Robust and Controllable Generative Models by Leveraging Physics-Based, Probabilistic, and Geometric Methods

by

Rajhans Singh

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved August 2023 by the
Graduate Supervisory Committee:

Pavan Turaga, Chair
Suren Jayasuriya
Visar Berisha
Pooyan Fazli

ARIZONA STATE UNIVERSITY

December 2023

ABSTRACT

Generative models are deep neural network-based models trained to learn the underlying distribution of a dataset. Once trained, these models can be used to sample novel data points from this distribution. Their impressive capabilities have been manifested in various generative tasks, encompassing areas like image-to-image translation, style transfer, image editing, and more.

One notable application of generative models is data augmentation, aimed at expanding and diversifying the training dataset to augment the performance of deep learning models for a downstream task. Generative models can be used to create new samples similar to the original data but with different variations and properties that are difficult to capture with traditional data augmentation techniques. However, the quality, diversity, and controllability of the shape and structure of the generated samples from these models are often directly proportional to the size and diversity of the training dataset. A more extensive and diverse training dataset allows the generative model to capture overall structures present in the data and generate more diverse and realistic-looking samples.

In this dissertation, I present innovative methods designed to enhance the robustness and controllability of generative models, drawing upon physics-based, probabilistic, and geometric techniques. These methods help improve the generalization and controllability of the generative model without necessarily relying on large training datasets. I enhance the robustness of generative models by integrating classical geometric moments for shape awareness and minimizing trainable parameters. Additionally, I employ non-parametric priors for the generative model's latent space through basic probability and optimization methods to improve the fidelity of interpolated images. I adopt a hybrid approach to address domain-specific challenges with limited data and controllability, combining physics-based rendering with generative models for more realistic results.

These approaches are particularly relevant in industrial settings, where the training datasets are small and class imbalance is common. Through extensive experiments on various datasets, I demonstrate the effectiveness of the proposed methods over conventional approaches.

ACKNOWLEDGMENTS

This dissertation is heartily dedicated to the invaluable individuals who have been integral to my journey. Without their unwavering guidance, encouragement, and support, the realization of this dissertation would not have been conceivable.

I am grateful to my primary Ph.D. advisor, Dr. Pavan Turaga. Their mentorship has been a beacon guiding my academic voyage, illuminating the path with their profound wisdom and insightful guidance. Their astute understanding of research intricacies and unwavering dedication to scholarly excellence has been nothing short of instrumental in shaping my doctoral journey. Moreover, their invaluable support extended beyond academics and into my professional development. For this, I am eternally grateful.

I sincerely thank Dr. Suren Jayasuriya, Dr. Visar Berisha, and Dr. Pooyan Fazli for serving as my committee members. Your helpful insights, feedback, and advice have significantly enhanced my dissertation. I am fortunate to have learned from your rich expertise, and I sincerely appreciate the time and effort you've invested in reviewing and steering my work. Your support has been pivotal to my journey.

I want to thank the senior members of the Geometric Media Lab (GML): Dr. Kuldeep Kulkarni, Dr. Suhas Lohit, Dr. Ankita Shukla, and Dr. Anirudh Som. Your guidance and help have been significant to me. I also want to thank my labmates, Dr. Kowshik Thopalli, Dr. Hongjun Choi, Eunsom Jeon, Sinjini Mitra, and Niccolò Meniconi. Our discussions, team projects, and friendships made my time in the lab a great experience. To all of you, I'm grateful. You've helped me learn and grow and made the lab a good place to work.

I want to say a big thank you to all my mentors at Intel and PNNL. I had great summer internships there. I learned a lot from Dr. Martin Braun, Dr. Ravi Garg, Dr. Nital Patel, Dr. Yehia Ibrahim, and Dr. Joon Yong. Working with them was a great experience. They helped me get the necessary industry experience that I really appreciate.

Lastly, but most importantly, I want to say thank you to my parents, Gayatri Devi and Janmejay Singh. They have sacrificed so much and always believed in me. I'm so grateful for their love and support. I also want to thank all my siblings and other family members. Their encouragement has always pushed me forward. I couldn't have done this without all of you.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Chapter 1

INTRODUCTION

With the rise of deep neural networks, generative models have become a trending topic in recent years [73, 42, 25]. They're used for generating different data modalities like text [90], graphs[170], speech [151], images [25], and videos [5]. In simple terms, these models take a low dimensional 'noise' vector from a known distribution and transform it into a complex high dimensional data point, matching the distribution of the training dataset. Generative models are very effective in various computer vision tasks such as text into images [121], image-to-image translation [120], domain adaptation [54], image editing [173], and inverse problems [129].

However, it is worth noting that the performance of these models in terms of quality, diversity, and control over the generated shapes and structures often hinges on the size and diversity of the training data. The more comprehensive and varied the training data, the better these models can understand the structures within the data and thus create more varied and realistic samples. In this context, my dissertation, titled 'Robust and Controllable Generative Models by Leveraging Physics-Based, Probabilistic, and Geometric Methods,' presents methods to enhance the robustness and control of generative models, mainly when working with smaller datasets. The proposed techniques include a hybrid approach that combines physics-based simulators with deep generative models, and the improved design of the latent space to increase image fidelity for interpolation. I also explore shape-aware and interpretable generator and discriminator architectures, drawing inspiration from geometric moments. These approaches make generative models more robust and easier to control. In the following sub-sections, I will delve into each contribution of this dissertation, explaining them in more detail:

1.1 A Hybrid Physics and Deep-learning Based Approach

Deep learning models have achieved state-of-the-art performance in classification [49] and detection tasks [115, 48]. However, they require large, high-quality training datasets that capture all the variations and nuances that could emerge during deployment. Collecting such extensive datasets, especially in industrial fields like biomedicine or manufacturing, can be exceedingly challenging. Industrial applications often grapple with small size and class imbalance datasets.

Deep generative models are often used to generate synthetic images for training sets in these challenging situations. However, for these models to generalize well, they too, require large datasets. Additionally, the controllability of the generated images from these models and knowledge transfer from other domains is often challenging [131]. This dissertation proposes a hybrid approach to tackle the challenge of small datasets and controllability. The proposed method uses a physics-based 3D rendering tool to model specific aspects of an image, combined with a small generative model to refine the rendered image into a realistic one. This method, which has a small set of easily controllable parameters, requires fewer training samples and offers enhanced control. The proposed approach has been tested in a real-world context—creating synthetic images of defects in Intel-manufactured chips. This dissertation provides qualitative and quantitative results that show the effectiveness of this model in limited training data scenarios.

1.2 Non-parametric Prior for Latent Space

Interpolation plays a vital role in generative models [110]. When we perform interpolation in the latent space of generative models, it results in meaningful interpolation in the image space. Interpolation confirms that the generative models are not merely memorizing the training data but generating novel samples from the distribution. Despite this, a common assumption in most generative model designs is using a simple parametric distribution, like Gaussian or Uniform, in the latent space. While this simplifies the implementation of the generative model, it can present challenges during interpolation due to distribution mismatches [155], i.e., the interpolated points distribution is different from the prior distribution of the latent space. The generative model is trained to generate realistic samples from the prior points; however, during interpolation, the path often traverses less dense regions of the latent space, decreasing the fidelity of the generated image [155, 71, 2, 82].

This dissertation uses a simple approach to understanding and solving this distribution mismatch issue. By using basic probability theory and readily available optimization tools, I develop ways to arrive at appropriate non-parametric priors. The obtained prior exhibits unusual qualitative properties in terms of its shape, and quantitative benefits in terms of lower divergence between the prior and interpolation point distribution. This part of the dissertation demonstrates that our designed prior helps improve image generation along any Euclidean straight line during interpolation, both qualitatively and quantitatively, without any additional training data or architectural modifications. The proposed formulation is quite flexible, paving the way to impose newer constraints on the latent-space statistics.

1.3 Geometric Moments Based Discriminative Model

The discriminator network plays a crucial role in the training of generative models. Its various functions make it indispensable - for instance, in Generative Adversarial Networks (GANs) [41], it distinguishes between real and fake images. The discriminator network is also utilized for conditioning purposes [99, 25], enabling the generation of images across a wide variety of classes. Importantly, it is from this discriminator that the generator learns, further emphasizing the critical role the discriminator network plays within the framework of generative models. Most discriminator networks are based on convolutional neural networks. It is well known that the CNN-based network tends to rely heavily on texture information rather than object shape for discriminative tasks [39]. Efforts have been made to develop deep models that are more aware of shape. However, it often proves challenging to create these models in a way that is straightforward, interpretable, and grounded in established mathematical definitions of shape.

This dissertation introduces a deep-learning model inspired by geometric moments, a classic, well-understood approach to measure shape-related properties [57, 70, 3, 32]. The proposed method consists of a trainable network for generating coordinate bases and affine parameters for making the features geometrically invariant yet in a task-specific manner. The proposed model improves the final feature's interpretation. This work demonstrates the effectiveness of the geometric moments on standard image classification datasets. The proposed model achieves higher classification performance compared to the baseline and standard ResNet [49] models while substantially improving interpretability.

1.4 Geometric Moments Based Generator Model

Building on using geometric moments for discriminative tasks, I further expanded this idea to encompass image generation tasks. In geometric moments literature [57, 143, 55, 33], reconstructing an image from its moments consider the image a polynomial function of its coordinate location, with the coefficients of this polynomial computed from the image's moments. This formulation can represent an image as a continuous function of its pixel location. Interestingly, neural networks are often used to approximate this continuous function. This type of representation is referred to as Implicit Neural Representation [98, 133].

Implicit neural representations (INR) have gained significant popularity for signal and image representation for many end-tasks, such as inverse problems [133, 112], 3D modeling [98], and more. Most previous INR architectures rely on sinusoidal positional encoding, which accounts for high-frequency information in data. However, the finite encoding size restricts the model's representational power. Higher representational power is needed to go from representing a single given image to representing large and diverse datasets. This dissertation addresses this gap by representing an image with a polynomial function and eliminates the need for positional encodings. Therefore, to achieve a progressively higher degree of polynomial representation, I use element-wise multiplications between features and affine-transformed coordinate locations after every ReLU layer. One significant benefit of representing an image in polynomial form is its inherent separation between shape and style. In this representation, the coefficients of lower-order polynomials typically capture shape-related information. Conversely, coefficients of higher-order polynomials convey style-related aspects. The proposed method is evaluated qualitatively and quantitatively on large datasets like ImageNet [22]. The proposed Poly-INR model performs comparably to state-of-the-art generative models without any convolution, normalization, or self-attention layers, and with far fewer trainable parameters. With much fewer training parameters and higher representative power, this approach paves the way for the broader adoption of INR models for generative modeling tasks in complex domains.

Chapter 2

GENERATIVE ADVERSARIAL NETWORK

2.1 Introduction

Generative models are useful tools that help us model complex data distribution and create new samples from this distribution. There are various types of generative models, such as Variational Autoencoder (VAE) [73], Flow-based model [26], Generative Adversarial Network (GAN) [41], and Diffusion model [158]. VAE is based on an auto-encoder constraining the latent space to a Gaussian distribution. Flow-based model is an invertible network in which the same parameters are used to define both the encoder and the decoder. GAN and Diffusion models are different because they do not have an encoding network. GAN works by having two neural networks compete against each other. In contrast, the Diffusion model generates data points by taking a random path between the latent space and data point in an iterative manner.

Generative Adversarial Network (GAN) is particularly exciting and forms a central theme of this dissertation. First introduced by Ian Goodfellow et al. 2014, GAN gained much attention primarily for its effectiveness in generating more realistic samples than auto-encoder-based models. A GAN architecture consists of a generator G and a discriminator D. The generator G maps low-dimensional latent points $z \sim \mathcal{P}_z$ to high-dimensional data distribution $\mathcal{P}_x$. The goal of the generator G is to produce data such that they are perceptually indistinguishable from real data. However, the discriminator D is trained to distinguish between 'fake' and 'real' data. Both the generator and discriminator are trained in an adversarial fashion, and at the end of the training, the generator learns to generate data with a distribution similar to the real one.

GANs are used in many areas, including text [171], images [68], videos [138], time-series data [84], 3D data [86], and graphs [152]. Particularly in computer vision, GANs have excelled in numerous tasks such as image-to-image translation [175], text-to-image conversion [126], image editing [4], and various inverse problems such as image inpainting [89]. There are different variations of GAN models for different tasks, demonstrating their adaptability and wide-ranging utility. To enhance our understanding of this powerful generative model, this chapter provides a comprehensive background on the inner workings of GAN. In the following sections of this chapter, we'll look more closely at how GAN works, training methods, metrics, and challenges.

2.2 Basic Concept and Theory

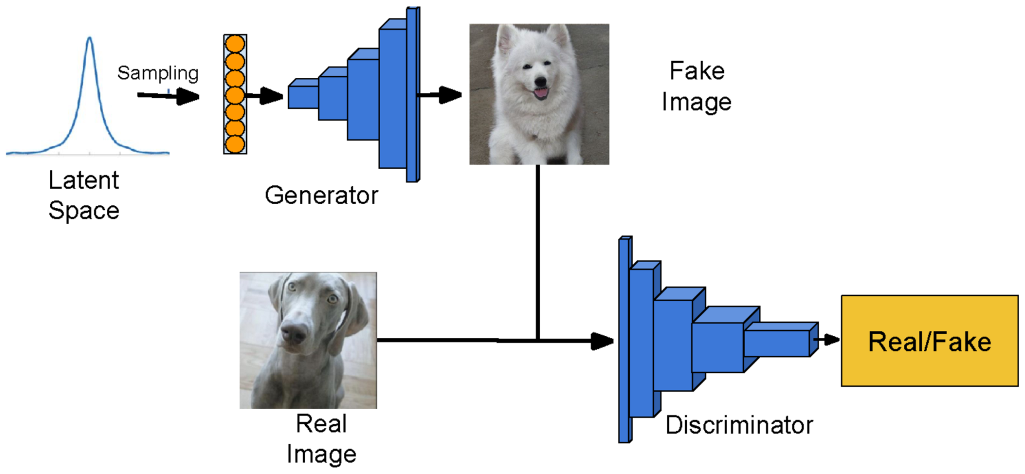


Figure 2.1: Block Diagram of a Generative Adversarial Network (GAN). This Model Is Composed of Two Core Components: a Generator Network, Tasked with Synthesizing Data, and a Discriminator Network, Designed to Distinguish Between Real and Synthesized Data.

The GAN model, as shown in Fig. 2.1, consists of two networks described as follows:

- **Generator:** The generator G produces synthetic samples when provided with latent vector z as input. This latent vector, drawn from a specific distribution ($z \sim \mathcal{P}_z$), introduces the necessary variability into the generated output. The latent-space distribution $\mathcal{P}_z$ is typically chosen to be a normal or uniform distribution. The goal of this generator is to learn and mimic the real data distribution. In essence, it is trained to fool the discriminator into assigning a high probability to its generated samples, indicating their resemblance to the real data.

- **Discriminator:** The discriminator D functions as a classifier that estimates the likelihood of a given sample coming from the real dataset $\mathcal{P}_{data}$ versus the synthetic dataset $\mathcal{P}_{G(z)}$. The discriminator operates much like a critic and is optimized to discern the difference between real and synthetic samples.

At its core, the discriminator network is essentially a binary classifier. For real data samples $x \sim \mathcal{P}_x$, the discriminator attempts to maximize the $E_{x\sim\mathcal{P}_x}[\log(D(x))]$. On the other hand, for generated samples $G(z)$, the discriminator's objective is to bring the value of $\log(D(G(z)))$ close to zero. This is achieved by maximizing the expected value of $E_{z\sim\mathcal{P}_z}[\log(1 - D(G(z)))]$. Conversely, the generator is trained to generate samples that fool the discriminator, i.e., to maximize the value of $E_{z\sim\mathcal{P}_z}[\log(D(G(z)))]$. Formally, this loss function can be expressed as a min-max game in (2.1), which the generator tries to minimize and the discriminator tries to maximize:

$$\min_G \max_D V(D,G) = E_{x\sim\mathcal{P}_x}[\log(D(x))] + E_{z\sim\mathcal{P}_z}[\log(1 - D(G(z)))], \tag{2.1}$$

The training of GAN involves simultaneous optimization of the generator and discriminator networks. This encourages both models to progressively improve and adapt to each other's performance. Every iteration performs a gradient step via backpropagation to minimize each network's cost function, thus optimizing their trainable parameters. It is shown by Goodfellow et al. [41] that the generator effectively minimizes the Jensen-Shannon divergence between the real and synthetic distribution for optimal discriminator:

$$\min_{G} V(D^*, G) = 2D_{JS}(\mathcal{P}_x || \mathcal{P}_{G(z)}) - 2\log(2) \tag{2.2}$$

where $D_{JS}(\mathcal{P}_x || \mathcal{P}_{G(z)})$ is given by:

$$D_{JS}(\mathcal{P}_x || \mathcal{P}_{G(z)}) = \frac{1}{2} D_{KL}(\mathcal{P}_x || \frac{\mathcal{P}_x + \mathcal{P}_{G(z)}}{2} + \frac{1}{2} D_{KL}(\mathcal{P}_{G(z)} || \frac{\mathcal{P}_x + \mathcal{P}_{G(z)}}{2} \tag{2.3}$$

where D_{KL} is the KL-divergence. Huszar et al. [59] emphasized that a key factor in GAN's success is the use of the symmetric Jensen-Shannon (JS) loss function in contrast to the asymmetric Kullback-Leibler (KL) divergence loss function commonly used in VAE models. In this training setting of GAN, when both the generator and the discriminator are optimally trained, we have $\mathcal{P}_{G(z)} = \mathcal{P}_x$.

Conditional GAN: Among the various variants of GANs, one particularly notable version is the conditional GAN (cGAN) [99]. In this model, the generator and discriminator are 'conditioned' on additional information, such as a class label 'y'. The class label y is fed to both generator and discriminator using additional input layers. This additional conditioning allows the cGAN to generate data corresponding to a specific class, providing enhanced control over the nature of the generated samples. GAN loss for the conditional version becomes like this:

$$\min_{G} \max_{D} V(D, G) = E_{x \sim \mathcal{P}_x}[\log(D(x|y))] + E_{z \sim \mathcal{P}_z}[\log(1 - D(G(z, y)|y))], \tag{2.4}$$

Loss functions: The original loss function of cross entropy proposed in the [41], often led to vanishing gradient and instability during training. Recognizing these challenges, researchers over the years have proposed a variety of loss functions designed to improve both the stability of training and the quality of the generated data.

- *Least Squares GAN* (LSGAN) [95] uses the least squares loss function, addressing the problem of vanishing gradients. This function also helps LSGANs generate higher-quality images and stable training. The least-square loss function is given by:

$$\begin{aligned} \min_D V(D) &= \frac{1}{2}E_{x\sim\mathcal{P}_x}[(D(x)-a)^2] + \frac{1}{2}E_{z\sim\mathcal{P}_z}[(D(G(z))-b)^2], \\ \min_G V(G) &= \frac{1}{2}E_{z\sim\mathcal{P}_z}[(D(G(z))-c)^2] \end{aligned} \tag{2.5}$$

where a, b, and c are hyper-parameters.

- *Maximum Mean Discrepancy* (MMD), used in moment matching GANs [83], uses kernel maximum mean discrepancy to determine the distribution distances between real and fake samples. Generative Moment Matching Networks (GMMNs) deviate from conventional GANs. Instead of employing a discriminator network, they utilize a kernel maximum mean discrepancy-based two-sample test, with the kernel being learned in an adversarial manner.
- The *Wasserstein loss function*, as proposed in Wasserstein GAN (WGAN) [8], provides a robust and smooth estimate of the distance between the real and generated data distributions using optimal transport. In WGAN, the author highlights an issue with Vanilla GAN: when the discriminator becomes exceptionally strong (i.e., D(x) = 1 for real data and D(G(z)) = 0 for generated data), the model can face a vanishing gradient problem at these extreme points. To counteract this, WGAN introduces a critic network that assigns realness scores to the generated samples, effectively replacing the discriminator. The key to WGAN's innovation is the Wasserstein loss function, which helps linearize the loss function landscape, enabling the critic network to achieve optimal performance with fewer iterations. The Wasserstein distance, also known as the Earth Mover's distance (EM distance), originates from calculating the minimum cost required to transform one probability distribution into another - akin to reshaping a mound of earth. The Wasserstein distance between two distributions

$\mathcal{P}_r$ and $\mathcal{P}_g$ is given by:

$$W(\mathcal{P}_r, \mathcal{P}_g) = \inf_{\gamma \in \prod(\mathcal{P}_r, \mathcal{P}_g)} E_{x,y \in \gamma}[||x - y||], \tag{2.6}$$

where $\prod(\mathcal{P}_r, \mathcal{P}_g)$ is set of all joint distributions whose marginal distributions are $\mathcal{P}_r$ and $\mathcal{P}_g$. The above loss function is intractable due to infimum over $\prod(\mathcal{P}_r, \mathcal{P}_g)$. In the WGAN paper, the author makes it tractable by using Kantorovich-Rubinstein duality:

$$W(\mathcal{P}_r, \mathcal{P}_g) = \sup_{||f||_L \leq 1} E_{x \in \mathcal{P}_r}[f(x)] - E_{y \in \mathcal{P}_g}[f(y)], \tag{2.7}$$

where f is a critic network defined by a neural network with the constraints that f is a 1-Lipschitz continuous function. In [8], Lipschitz continuity constraints are imposed on the critic by clamping the parameters' weights to a small window such as $[-0.01, 0.01]$. Gulrajani et al. 2017 [44] pointed out that weight clipping is a bad way to enforce a Lipschitz constraint. Choosing a large clamping window leads to exploding gradient, and choosing a very small window leads to slow convergence. For these reasons, [44] proposes a gradient penalty term for the Lipschitz constraint. However, this gradient penalty term is computationally expensive. This is further improved using spectral normalization by [100]. The Spectral Normalization technique stabilizes training by normalizing the weights of each layer (weight matrix) with its corresponding spectral norm, also known as the matrix norm or the maximum singular value of a matrix. This enables the normalization of weights every time they are updated, creating a network that mitigates issues of gradient explosion and thus reduces instability during training. This method has shown comparable results to those achieved by gradient penalty but with improved computational efficiency.

- The Hinge Loss function, introduced in the Geometric GAN [88], is inspired by the Support Vector Machines (SVMs) theory and the concept of maximizing the margin of a separating hyperplane. This loss function contributes significantly to the stability of the GAN's training process and promotes better convergence. Hinge loss function is given by:

$$V(D, G) = E_{x \sim \mathcal{P}_x}[max(0, 1 - D(x))] + E_{z \sim \mathcal{P}_z}[max(0, 1 + D(G(z)))], \tag{2.8}$$

Thus, over time, the development of these alternative loss functions has significantly improved the capabilities of GAN. The choice of loss function directly impacts the quality of the generated samples and the stability of training, making it a critical consideration when working with GAN. Currently, Wasserstein-based GAN with hinge loss function and spectral normalization is the most popular choice.

Architecture: Over the years, GAN has been marked by significant architectural modifications that enhanced their performance and application. Initially, GAN, as introduced by Ian Goodfellow et al. [41], primarily consisted of fully connected layers. As CNNs became increasingly prominent in computer vision, Radford et al. [110] proposed a novel architecture for the generator and discriminator networks. This architecture, based on transposed convolution layers, is known as DCGAN. The advent of DCGAN marked a significant improvement in the performance of GAN. The DCGAN architecture is further improved over the year. Currently, the most famous architecture is Style-GAN [68]. Style-GAN's generator consists of a mapping network and a synthesis network. The mapping network consists of a multilayer perceptron (MLP) network that transforms the latent vector into style codes. The synthesis network consists of novel Adaptive Instance Normalization layers that take style codes as input and modulate the weights of the Conv layers. This architecture not only improved the performance of GAN significantly but also provided control over manipulating generated samples in terms of style. Most recently, Transformer-based models [150] have been adapted for GAN [165, 63]. These models leverage attention mechanisms to capture long-range dependencies within the sample. Transformer provides the benefits of huge learning capacity of the network and trains on one scale and samples at any scale feature.

2.3 Evaluation Metrics

There are many metrics to quantify the performance of a GAN model, and each metric has its own strengths and weaknesses.

Inception score: The inception score *correlates* with the visual quality of the generated image – higher the better. The Inception Score, proposed by Salimans et al. 2016 [122], evaluates the quality and variety of generated images. It is computed by passing these images through a pre-trained Inception model and evaluating $P(y|x)$. The resulting conditional label distribution $P(y|x)$ for a well-performing model should be highly specific, indicating that the image contains distinct, identifiable objects. The overall label distribution P(y) across all generated samples should show significant diversity. The Inception Score is the Kullback-Leibler divergence between $P(y|x)$ and $P(y)$.

Frechet Inception Distance: Recent studies suggest that the inception score does not compare the statistics of the generated dataset with the real-world data [51, 166], and thus is not always a reliable indicator of visual quality. This drawback of the IS is overcome by the FID score [51], which compares the statistics of the generated data with the real data with respect to features. FID employs a pre-trained Inception model to extract features from real (x) and synthetic (g) images. It specifically uses the last pooling layer, just before output classification, to capture the features of an input image. This feature space is considered a continuous multivariate Gaussian distribution. From the extracted features, the Fréchet distance, also known as the Wasserstein-2 distance, is computed between the real and generated distributions. The lower the FID score, the more similar the two sets of images are, with zero indicating an exact match. FID score is given by:

$$FID = ||\mu_x - \mu_g||^2 + Tr(\Sigma_x + \Sigma_g - 2(\Sigma_x \Sigma_g)^{\frac{1}{2}}) \tag{2.9}$$

where μ is the mean and Σ is the covariance of the features. Tr is the trace operation, i.e., the sum of the elements along the main diagonal of the square matrix.

Spatial Frechet Inception Distance: sFID [102] enhances the concept of FID by incorporating higher spatial features from the Inception network. This inclusion aims to account for the spatial structure of the generated image, providing a more comprehensive evaluation of the visual quality.

Precision and Recall: From the classic viewpoint, precision denotes the fraction of generated images that are realistic, and recall measures the fraction of the training data manifold covered by the generator [78]. In the context of GANs, Precision and Recall are computed using a pretrained classifier. The classifier's (inception network) features are used to construct a Gaussian distribution for both the real and generated data. Precision and Recall are then computed based on the overlap between these two distributions, using their means and covariance matrices.

2.4 Challenges

Despite the impressive capability of GAN to produce highly realistic images, significant challenges persist with these models:

- **Training instability:** Training GAN is notably challenging due to its inherent dynamic adversarial nature, which often leads to instability. Such instability can result from issues such as vanishing or exploding gradients. Thus, to ensure stable training, the selection of generator and discriminator architectures, loss functions, and normalization methods must be done carefully.

- **Mode collapse:** A prevalent issue known as Mode Collapse, where the generator repeatedly produces identical outputs, may arise during the training. Despite the ability to trick the corresponding discriminator, this indicates a failure of the generator to adequately represent the entire real data distribution, causing it to become entrapped in a limited space with minimal variety.
- **Lack of control:** Traditional GAN generates random samples without any control over the type or characteristics of the generated data, i.e., the generated samples are represented in an entangled and non-meaningful manner in the latent space. Although Conditional GAN and other variants offer more control, these models have challenges, such as having labeled information in the data.
- **Large training dataset and domain knowledge:** GAN models generalize better only if we have a large training dataset containing all variations. The GAN model trained on a significantly smaller training set generates images with less variation and often suffers from mode collapse. Furthermore, integrating real-world knowledge regarding objects such as shape, size, and structural variations in the GAN model is only possible with a substantial training set representing such variations.

This dissertation addresses these challenges by introducing novel methodologies aimed at boosting the robustness and controllability of the GAN models. These methods leverage techniques grounded in physics, probability, and geometry to improve GAN's performance.

Chapter 3

A HYBRID PHYSICS AND DEEP-LEARNING BASED APPROACH FOR SYNTHETIC DEFECT GENERATION

3.1 Introduction

With the complexity of semiconductor design and manufacturing processes increasing rapidly, quickly identifying defects induced by these complex processes has become a significant challenge. While human operators can be trained to detect defects through visual inspection, it becomes a rate-limiting factor in high-volume manufacturing processes. Thus, an automated inspection pipeline using computer vision and machine learning algorithms, is important for cost-effective deployments.

Due to recent advances in deep learning in computer vision, [79], deep learning methods have started to replace classical image processing and machine learning-based methods [45] in automated visual inspection systems. Deep learning models have achieved SOTA performance on classification [49], object detection [115] and semantic segmentation [14], making them a robust tool for integration in visual inspection systems. Recently, these methods have been used for defect detection in various industries [11]; for example, a light-weight deep learning model for electronic component detection [58], a convolution neural network based segmentation model to detect surface anomalies [140], and defect detection in printed circuit boards [1].

The design and development of automated visual inspection systems necessarily requires creating large training-sets, which need specific domain knowledge from an engineer or technician to label the variety of defects that can occur during the manufacturing process [131]. With the advent of deep learning, attaining the target performance requires larger training sets than the traditional vision algorithms that use hand-crafted features. The training datasets need to be curated and designed to encompass all variations and nuisance factors that may occur in the manufacturing process beforehand. Aside from the images acquired from production or development product runs, the training datasets for defects are often created in a destructive method, i.e., artificially creating defective samples in a lab beforehand. These destructive methods often involve time, cost and manual labor. Creating all kinds of defects that can happen under complex manufacturing processes and look realistic is also challenging. However, recent interest has shifted to developing synthetic defect generation approaches, to augment and speed-up the process of creating such training-sets as well.

Generative adversarial networks (GAN) [41] have been widely adopted as a data augmentation method to help augment small datasets or resolve class imbalance problems. A GAN generally consists of a generator and a discriminator. The generator takes a noise vector sampled from a distribution [132, 41] as input and generates a realistic-looking image. The discriminator takes real and synthetic images as input and classifies whether the input image is real or fake. The generator is trained to generate realistic-looking images such that it can fool the discriminator. GANs have been widely adapted in computer vision for various tasks like image-to-image translation [61], image super-resolution [80], image editing [174], and domain adaptation [130]. It is also been used to address class-imbalance problems [123] and as a data augmentation tool in defect generation [154, 101, 30, 10].

While synthetic images sampled from a GAN can help improve defect detection and classification performance, training a GAN model is a data-intensive task. It is a chicken and egg situation, in the sense, in order to make a GAN model synthesize all variations of defects, the training dataset must include all defect variation types, which defeats the purpose of using GAN in scarce data scenarios. It is also difficult to control parameters like the shape and size of a defect generated by a GAN, since it is not easy to isolate dimensions in the latent space that control physically meaningful attributes i.e., the defects are not represented in a disentanglement manner in the input latent space. The GAN model also finds it challenging to transfer knowledge from other domains, such as the visual appearance of foreign material or cracks, from other industrial products to semiconductors. This kind of knowledge can only be induced in a GAN through training datasets which are challenging to collect to begin with.

In this work, we propose a hybrid approach for synthetic defect generation. We model the defects in a 3D physics-based renderer such as POV-Ray [108], using a few controllable parameters which control the shape and size of defects. For example, as a first step we model the shape of Foreign Material as a spline curve and the shape of Epoxy defects as a polygon. We then choose convenient material reflectance properties to make the rendered defect look sufficiently close to being realistic from the available material choices. As a final step, we use a GAN to make these rendered defects look realistic, based on a small training set of realistic defects. In contrast to approaches that use only a GAN to generate defects from scratch, our model provides the shape and structure of defects by using a renderer, and only learns to apply realistic-looking texture through a small GAN model. Texture generation on the rendered images is a much easier task and requires very few training images to train the generator.

The key highlights of our work are summarized below:

- We propose a hybrid model that combines a 3D rendering tool with a GAN model. The proposed approach eliminates the need for large training datasets and provides ease of controlled defect generation, that saves cost and speeds up getting ROI in the product/process launch.

- We assess the quality of synthetic defects from our hybrid model by training a binary classification model on the augmented dataset. The classification model trained with our synthetically augmented dataset shows improved performance over the vanilla training dataset.
- We qualitatively and quantitatively evaluate the synthetic defects generated by two GAN models: style-transfer GAN and cycle-consistency GAN.
- Lastly, we do an ablation study of the classification model by varying the number of real images used along with the synthetic defects images for model training. Our synthetically generated defects achieves significant boost in the classification performance when the number of real defects in the training set is low.

3.2 Problem Setup

3.2.1 Dataset Details

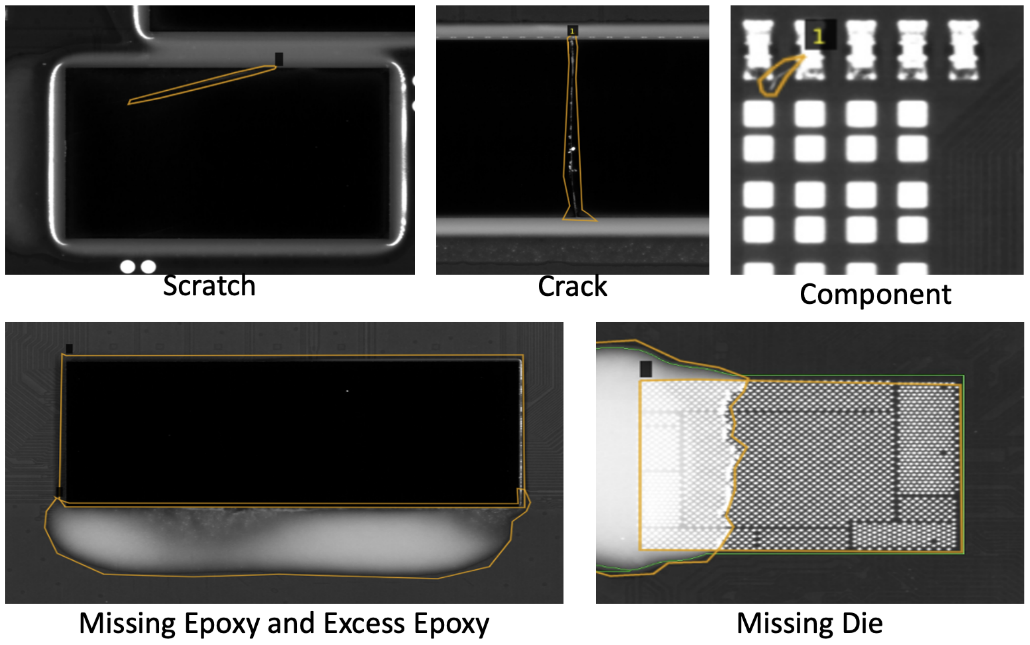


Figure 3.1: Examples of Type of Defects Present in Our Dataset.

Table 3.1: Number of Images per Defect Type in Training and Test Datasets.

Defect type	Training set	Test set
Foreign material	115	47
Component	16	6
Crack	18	8
Excess epoxy	11	4
Missing epoxy	20	5
Epoxy on die	13	4
Scratch	56	21
Missing Die	7	3
Other	139	31

We use a dataset that consists of 9 types of defects, namely *foreign material, component, crack, excess epoxy, missing epoxy, epoxy on a die, scratch, missing die, and others.* Figure 3.1 shows examples of some defects across these categories. The 'other' defect type consists of defects that cannot be classified into one of the 8 classes and includes rarely occurring defects. Table 3.1 shows the number of samples per defect type present in the training and test datasets. We divide the data into training and test sets in a ratio of $\sim 70\%$ and $\sim 30\%$, respectively. In this problem setup, we face three challenges:

- The number of samples per defect type in the training set is much smaller than any conventional natural image dataset used in deep learning literature. Deep learning model (classification or detection) trained on a small dataset often overfits and fails to generalize.
- The dataset has a significant class-imbalance between different types of defects. For example, foreign material defect has significantly more images (115), whereas

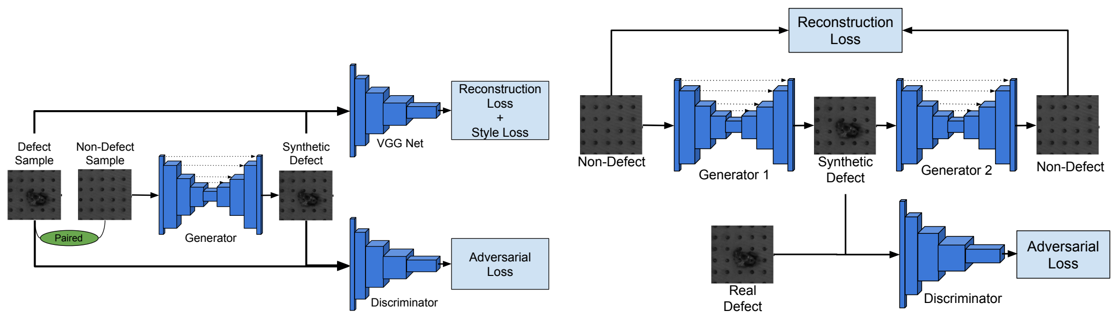
defect like missing die only has 7. Deep learning model trained on a class-imbalanced dataset becomes more biased towards the defect type present in large quantities, sometimes ignoring minority classes entirely.

- Lastly, higher resolution images are required to detect finer defects like scratches or foreign material. This dataset consists of images of approximately $2K \times 2K$ resolution, which is computationally expensive to use with any deep learning model. The most common deep learning models (classification, detection or generative) operate on either 256×256 or 512×512 size images.

We use a deep-learning based generative model to solve the data scarcity and class-imbalance issues. The generative model solves both problems by synthetically generating more samples for both minority and majority defect classes. We use a patch-based method to overcome the challenge of high resolution by extracting smaller patches (256×256 or 512×512) around the defects for classification, detection, or image generation.

3.3 Proposed Method

3.3.1 GAN Models for Synthetic Defects



(a) Style Transfer GAN Model (b) Cycle GAN Model

Figure 3.2: Network Architecture for (a) Style Transfer GAN and (b) Cycle GAN Model for Defect Generation.

Datasets obtained from high volume manufacturing processes tend to be imbalanced with far more non-defect samples than defective ones. Therefore, to address the need to create effective training-sets, we use an image-to-image translation method to transform non-defect images to defect images using GAN models. We adopt two GAN models used in [131] to generate synthetic defects: 1) Style Transfer GAN and 2) Cycle Consistency GAN.

Style Transfer GAN: Figure 3.2a shows the block diagram of style transfer GAN. This model consists of a generator module with UNet [117] architecture, consisting of convolutional layers with batch normalization and ReLU layers. The generator takes a non-defect input patch and generates the defect on top while keeping the background unchanged. The model is trained with three loss functions: 1) Style transfer loss [38], 2) Perceptual loss [64]and 3) Adversarial loss [41]. Style transfer loss helps to generate defects with a similar texture as the real one. Perceptual loss is a reconstruction loss ($l2$ distance) with respect to the target defect image in pre-trained VGG feature space. Adversarial loss is based on a discriminator network which distinguishes whether the input sample is real or fake, helping in generating realistic-looking defects. The reconstruction loss in this model requires paired defect and non-defect training samples, i.e., the input non-defect sample must have the same background as the target defect sample. Images of the same semiconductor module often get misaligned due to vibration or other mechanical or optical noises during manufacturing. If the input images are not perfectly paired with the training targets, the generator outputs can be distorted during test time. We employ another image-to-image translation model called cycle consistency GAN to avoid the need for pairing datasets during training.

Cycle Consistency GAN: Figure 3.2b shows a block diagram of the cycle GAN model [175]. The cycle GAN consists of two generators with the same architecture as the style transfer GAN generator. Generator-1 takes a non-defect input patch and converts it into a defect image. Generator-2 then takes the defect image generated by generator-1 and converts it back to the original non-defect image. Both generators are trained with reconstruction loss ($l2$ norm) between the input non-defect image and output image of generator-2 and an adversarial loss on the generator-1 by using a discriminator network which forces the generator-1 to generate realistic-looking defects. This model does not require paired non-defect and target defect images for training. It automatically finds a one-to-one mapping between non-defect and defect samples, provided the dataset is large enough; otherwise this condition may lead to a mode collapse. The cycle GAN method can also generate different types of defects on the same input patch by feeding different defect label information to generator-1 [99].

3.3.2 Hybrid Model for Synthetic Defect

GAN models described in the above section need a significantly large training dataset to generate all variations of defects. In addition, we can only interpolate between defects present in the training set, and it is difficult to control the variation of generated defects or add prior knowledge about defects without having a large training dataset. These defects can be modeled effectively with physics-based rendering tools like POV-Ray with few controllable parameters. We model two common defects: Foreign Material and Epoxy defects, with a rendering tool and use the rendered defects with a small GAN model to refine them to realistic-looking defects. This hybrid model allows users to generate any variety of defects with a minimal training dataset.

Foreign Material Rendering:

Foreign Materials (FMs) in semiconductor manufacturing are usually fibers that originate from clean room smocks, metal shavings, human hair, etc. These fibers

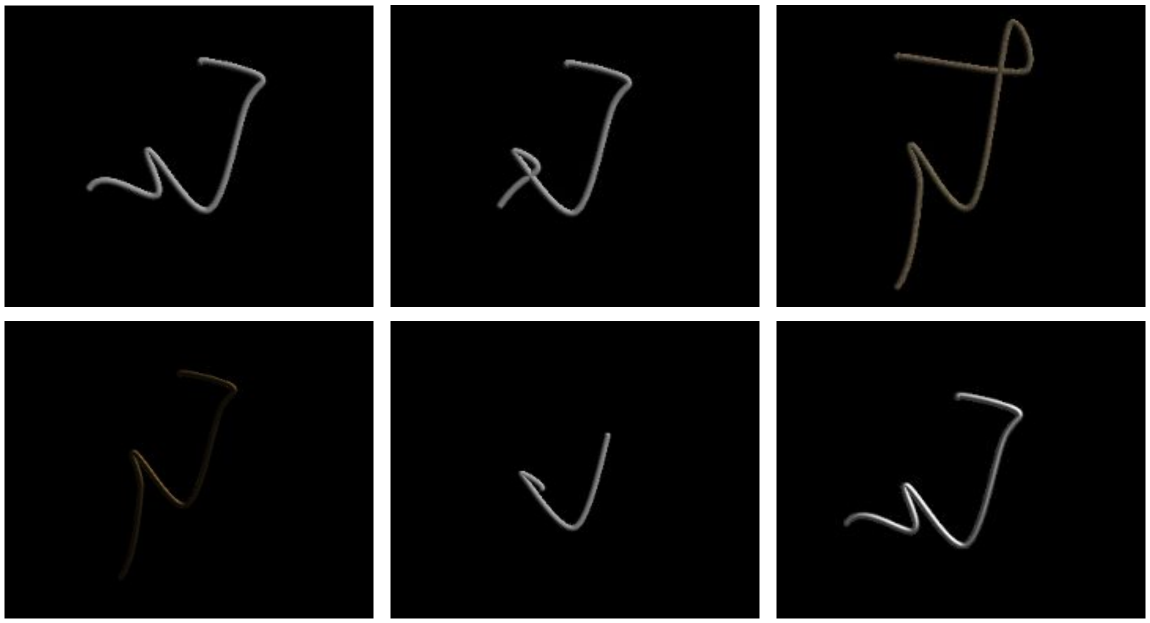


Figure 3.3: Example of Rendered FM Defects from Pov-Ray.

generally have a high aspect ratio and curvature that lends itself to being well-modeled by spline curves. Hence we model FM as a $3D$ spline curve defined by critical points and a polynomial fit. We use a 3^{rd} order polynomial in our experiments, and the number of critical points varies from 4 to 20, depending on the complexity of the spline curve. We use POV-Ray's python scripting language to define our 3D defect models. Modeling FM as a spline curve is advantageous as we can easily control the shape and size, and also interpolate between different curves to further increase control. Figure 3.3 shows examples of a few rendered FM defects from POV-Ray. Notice that we can render Foreign Material with various spline shapes and different reflectance properties.

However, synthetic FM defects created in this way do not help much in improving the performance of a classification model when used as part of an augmented training dataset. We validate this observation by performing a FM classification task using a ResNet-18 model trained on three datasets: only real FM images, only synthetic FM images with random spline curves, and only synthetic FM images with spline curves matching the real FMs (see Table 3.2, first column). In Table 3.2, we use real FM

Table 3.2: Performance Comparison of FM Classification Model Trained on a) Only Real b) Synthetic with Random Spline FM, c) Synthetic with Spline Shape Matching Real FM Images, and Tested on Real FM Images.

Training Images	Accuracy (%)
Real FM	95.74
Synthetic FM: Random Spline	61.82
Synthetic FM: Matching Spline	82.87

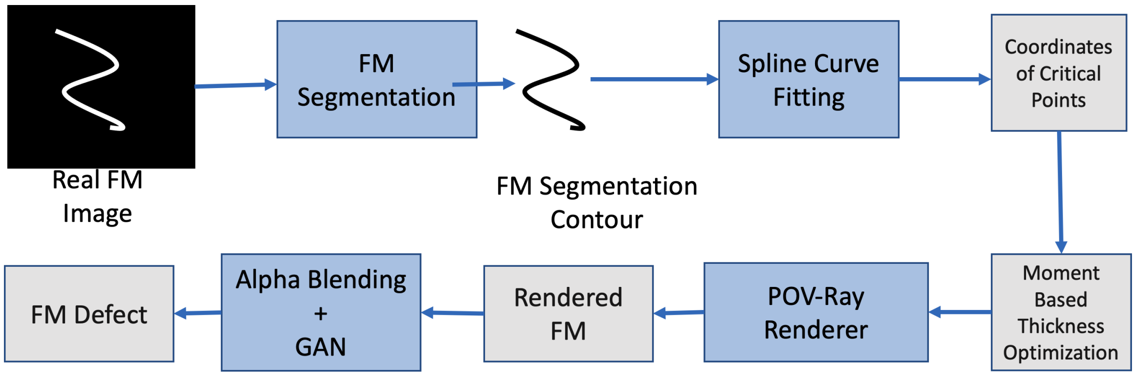


Figure 3.4: Overview of Our Spline Fitting Method for Synthetic FM Rendering.

defect images as test set. Table 3.2 shows that the classification model trained on the synthetic FM images defined by random spline curves performs worse than the other two models. We also observe that the performance is much better when the model is trained on more realistic-looking FM defects obtained by matching the spline shape to real FM, compared to the random spline images. Hence for generating synthetic FM images, it is essential to define the spline curves shape like the real FMs shape.

We use a spline fitting method to extract the distribution of real FM curves. Figure 3.4 shows the block diagram of our method. We first segment the defect region from the real FM defects in our method. We use a manually annotated segmentation mask around the FM, which is further refined by the Canny edge-based segmentation method. We then find the contour points along the segmented FM defect using the OpenCV's findContours function. We use the SciPy implementation of least-squares

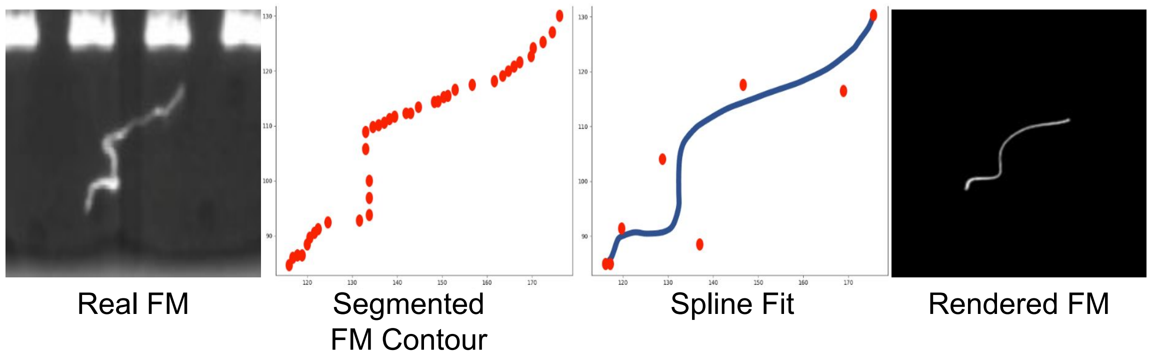


Figure 3.5: Example of Spline Fitting on a Real FM Image (Leftmost) and Final Rendered FM from POVRay.

based spline-fitting method to fit the spline along the contour points. FM defects in manufacturing can be found in various shapes and thicknesses. It is also essential to sample the thickness of these spline curves from the real FM thickness distribution. To find the correct thickness, we use the grid search method. We use geometric moments [57] as a feature to measure the similarity between the rendered FM and the real FM for the thickness grid search. Geometric moments are well-known shape descriptors in computer vision literature, and it is better than vanilla $L2$ distance as we want to avoid the variation in texture or intensity in the real FM image. Figure 3.5 shows an example of the spline fitting method on a real FM defect image (first) and final rendered FM (fourth). The rendered FM image from POV-Ray is first merged with a non-defect background image using the alpha blending method given by Eq-3.1:

$$I_{blend} = \alpha I_{rendered} + (1 - \alpha) I_{background} \tag{3.1}$$

where, $I_{rendered}$ is the rendered image, $I_{background}$ is the non-defect image and α is real value taken between 0 and 1. Figure 3.6 shows an example of alpha blending. After alpha blending, we use a small generator model consisting of a few convolutional neural network layers to further refine the blending with the background image. This small generator model takes the blended synthetic defect image and is trained with the reconstruction and adversarial loss to output realistic-looking defect images.

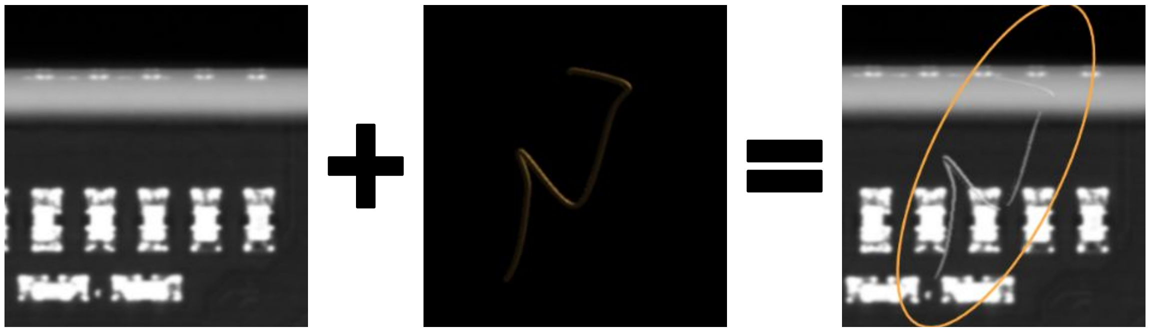


Figure 3.6: Example of Alpha Blending. Left Image Is the Non-defect Sample, Middle Image Is the Rendered FM Image, and Right Image Is the Blended Image.

Epoxy defect rendering:

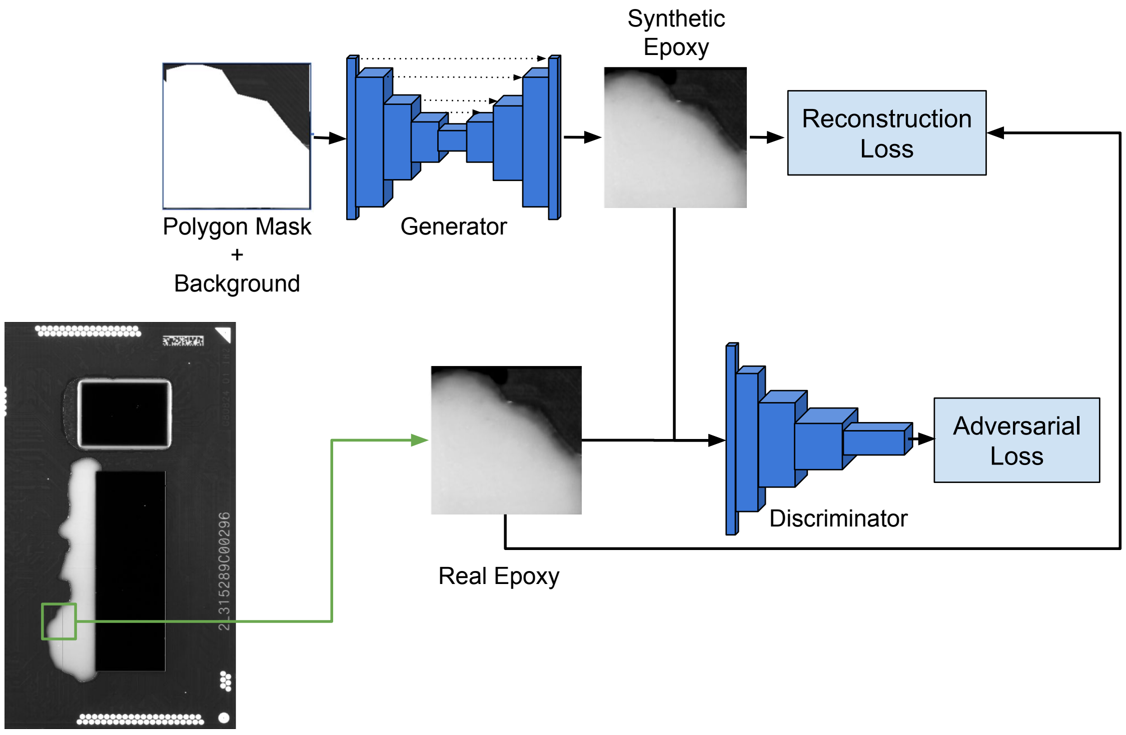


Figure 3.7: Overview of Our Epoxy Defect Rendering Model.

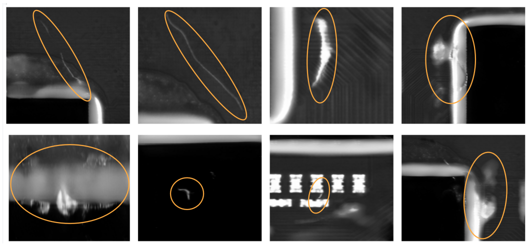
We model the Epoxy defect as a binary polygonal mask image. Unlike FM, the Epoxy defect is difficult to model in $3D$ with few easily controllable parameters because of the significant variation in shape and structure. As a result, in this case, we feed the polygonal mask and a background non-defect image to a generator directly. Figure 3.7 shows the block diagram of our Epoxy rendering method. The generator in this model is composed of several CNN layers and is trained with reconstruction loss ($l2$ norm) and adversarial loss with the help of a discriminator to generate realistic-looking Epoxy defects. In this method, we provide the shape of the Epoxy defect in the form of a polygonal mask, and the generator only generates a realistic-looking Epoxy texture on the mask region. This texture generation is much simpler than generating an entire defect from scratch, so only a few samples are required to train the generator. We sample our polygonal mask from the real Epoxy defect shape distribution by matching the mask to the real Epoxy defect images. We use a perturbed polygonal mask and a random non-defect image background during defect generation for dataset augmentation.

3.4 Experiments and Results

3.4.1 Synthetic Defects from GAN

Our experiments use the unconditional version of style transfer and the class-conditional version of cycle GAN models. Both models are trained to generate all defects types from Table 3.1. We train both style-transfer and cycle GAN on 512×512 patch images extracted randomly around the defect region of the training images. Both models are trained with a batch size of 16 and up to $500K$ iteration, which is long enough for the GAN to converge and generate realistic-looking defects. We use the Adam optimizer with learning rates of $1e^{-4}$ for the discriminator and $2e^{-4}$ for the generator. We use Wasserstein distance [8] as loss function and spectral normalization [100] for stability. In both GAN models, we use a discriminator of 6 ResNet layers [49].

The style GAN's generator with U-Net architecture consists of 12 ResNet layers with kernel size 3×3 and feature channels of 256. For style GAN we need perfectly aligned paired defect and non-defect training datasets having the same backgrounds. We use image registration to align the defect and non-defect samples before training. Cycle GAN generator is made up of 12 ResNet layers and is trained with adversarial and cycle consistency loss functions. In cycle GAN, we do not need the paired defect and non-defect images; hence we supply randomly extracted patches from the non-defect image as input.


(a) Style Transfer GAN

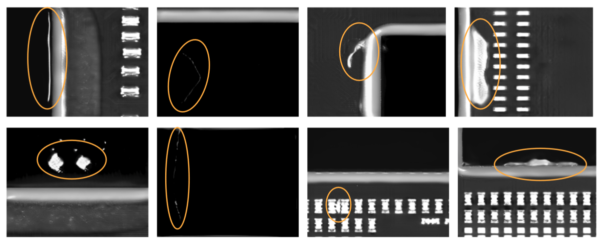

(b) Cycle GAN

Figure 3.8: Example of Defect Images Generated from (a) Style Transfer GAN and (b) Cycle GAN.

Qualitative Results

Figure 3.8a and 3.8b shows example of defects generated by style-transfer and cycle GAN respectively. We observe that both models can generate very realistic and diverse defects. Similar to [131], we also observe that the background texture of the generated defects in the style-transfer GAN is slightly distorted. This distortion is mainly due to the non-perfect alignment between the defect and non-defect images in the training dataset. Furthermore, the defects generated by the style-transfer GAN are slightly inferior (notice the more spread out in epoxy-related defects) compared to the cycle GAN. The figure shows that the cycle GAN can generate sharp defect images while preserving the background. However, cycle GAN generates better results qualitatively but converges slowly and requires more computation in training due to the training of two generator models compared to the single generator in style-transfer GAN.

Quantitative Results

We use a deep learning-based object detection model as a defect detection system to quantify the generated defects. We use the Faster RCNN model [115] to detect all defect types from the Table 3.1. This defect detection model generates the bounding box around the defects and classifies the defect types in the given input images. We train the defect detection model on three datasets: only real images, Style-transfer GAN augmented images (Real + Synthetic), and cycle GAN augmented images (Real + Synthetic). We evaluate the performance of these three models on the same real test dataset to quantify the quality of synthetically generated images from GAN.

We use the open-source TensorFlow implementation of the faster RCNN. We manually annotate the defect label and bounding box for the training and test dataset. For the generated images from the GAN model, we manually screen the synthetic defect images by removing the highly distorted defect images and manually annotate the bounding box around the defect. We use around 100 or more generated images for each defect type from the GAN models to equal the number of defect images per defect type. We use ResNet-50 model pre-trained on the ImageNet dataset [22] as backbone for the detection module. We train the model on a patch size of 512×512 cropped randomly around the defects, and the image size is scaled randomly by factors between 0.5 and 2.0 chosen uniformly. We train the model with a batch size of 16 and up to $20K$ iterations (long enough for all models to converge).

The mean Average Precision (mAP) metric is used to quantify the detected defects from the three models. The mAP metric is popular for measuring object detection in natural images. The detection model produces two results: a bounding box that contains the defect region and a probability score. The bounding box performance is measured using Intersection over Union ($IoU = \frac{AreaofOverlap}{AreaofUnion}$) with respect to the ground truth bounding box. Based on the IoU threshold, we can evaluate True Positive or False Positive predictions. Then average precision for a particular class is the area under $Precision = \frac{TruePositive}{TotalPredictedPositve}$ and $Recall = \frac{TruePositive}{TotalGroundTruthPositive}$ curve under different class probability score and mAP is given by $mAP = \frac{1}{N}\sum_{i=0}^{N} AP_i$, where AP_i is the average precision for defect type i.

Table 3.3: Performance Comparison of Defect Detection System Trained on Different Datasets.

mean Average Precision	Only real images	Style transfer GAN aug. dataset	Cycle GAN aug. dataset
IoU = 0.5:0.95	26.34	**28.79**	27.49
IoU = 0.5	45.13	**53.82**	47.15
IoU = 0.75	24.66	27.89	**28.23**

Table 3.3 shows the mAP of the three models on the test dataset. In the first row, the IoU threshold is set from 0.5 to 0.95 with an increment of 0.05 to compute the mAP. In the second and third rows, 0.5 and 0.75 are taken as thresholds for IoU. In the table, we observe that the models trained on the augmented images (real + synthetic) perform better than the model trained only on the real images. We see an improvement of at least 1.15% with augmented images. We also observe that the model trained with style transfer GAN augmented images performs better than the cycle GAN augmented images when the IoU threshold is chosen smaller ($IoU = 0.5$), but at a higher IoU threshold value ($IoU = 0.75$) cycle GAN augmented model performs better than the style transfer GAN augmented model. Table 3.4 shows the minimum training iterations required by the three models to converge to maximum test performance. We observe that the model trained on the augmented images converges faster than the model trained on only real images. Figure 3.9 shows examples of detected defects from the three models where the model trained on only real images failed to detect

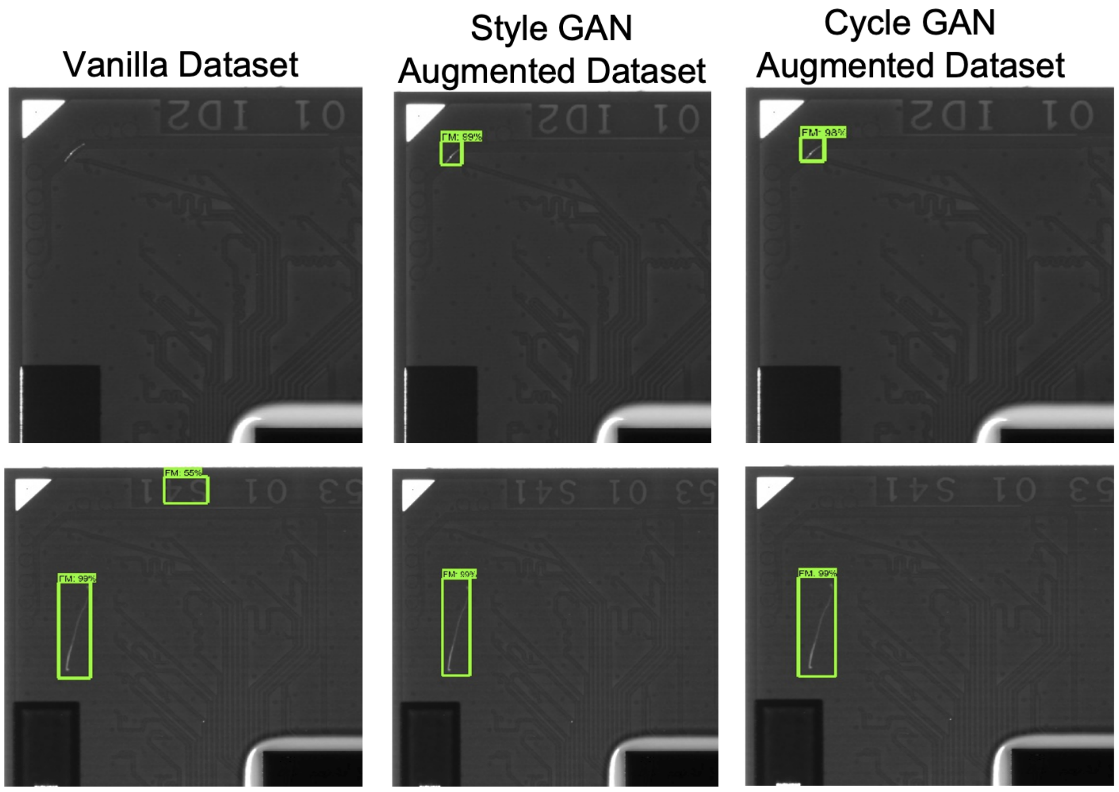


Figure 3.9: Example of Detected Defect from Our Detection Model Trained on Different Datasets.

(first row) and false detect (second row), whereas the GAN augmented models able to detect it correctly.

Table 3.4: Iteration Taken by Defect Detection Model Trained on Different Datasets to Converge to Maximum Test Performance.

Training Dataset	Training Iteration
Only real images	7.5K
Style transfer GAN Aug.	4.8K
Cycle GAN Aug.	5.2K

Limitation of GAN Models

As the previous section shows, GAN augmented images can boost the defect detection system's performance. On the other hand, GAN models generalize better only if we have a large training dataset containing all variations of the defects, but collecting such large datasets is expensive and time-consuming. The GAN model trained on defect types with a significantly smaller training set generates images with less variation. Also, if a completely different non-defect patch is provided as input, which is different from the training set, the GAN models distort the background texture of the generated defects. Using these distorted defect images to train a defect detection system degrades the performance, as compared to detection model trained only on the real images. As a result, manual screening is needed to remove these distorted synthetic defect images before using them to train the detection model.

Furthermore, GAN models are widely known for their unstable training and require hyper-parameters fine-tuning for stable training. The GAN model can only generate defects sampled from the distribution of real defects in the training set. Integrating real-world knowledge regarding defects such as shape, size, and structural variations in the GAN model is challenging without a substantial training set representing such variations. For example, a human can comprehend the variations occurring in foreign materials based on their world knowledge but controlling those variations in a GAN model without an extensive training set is a challenging problem.

3.4.2 Synthetic Defects from Rendering

This section discusses the experiment setup and results regarding our hybrid model for two defects: Foreign Material and Epoxy-related defects (excess Epoxy and Epoxy on die).

FM Rendering

As explained in the method section, we use spline fitting and geometric moment-based optimization for thickness grid search to get the distribution of Foreign Material's shape and thickness from the training set. One can also find this distribution from other products or domains. We render $1K$ synthetic Foreign Material images from this distribution by perturbing the critical point of the spline and thickness with a Gaussian distribution. We use the alpha blending method with an alpha value of 0.3 to combine the rendered spline with the non-defect images. The generator model in this hybrid approach consists of a 4 CNN layer with batch normalization and ReLu layer. The number of feature channels is 256 in each layer. It takes input image of 256×256 and generates image of size 256×256. To train this model, we created a dataset consisting of paired, rendered FM images from PovRay and Real FM images with the same background. We train this model with a batch size of 16 up to $30K$ iteration and a learning rate of $2e^{-4}$ and Adam optimizer. Once trained, we can use this generator to combine any rendered FM defects with any non-defect background image. Figure 3.10 shows a few example of final rendered FM images. Unlike the previous standard GAN models, the hybrid model generates defects with realistic-looking shapes and no artifacts around them. Hence, no manual screening is needed when the synthetic defects are used as data augmentation for the detection or classification model training.

We use a binary classification model to quantify the quality of generated images from this hybrid model. The classification model is trained with the synthetic augmented dataset and tested on real FM images. We created a dataset by randomly cropping images of size 256×256 around the FM defects from the training set. We use the ResNet-18 model for classification. We use the batch size of 32 with half images

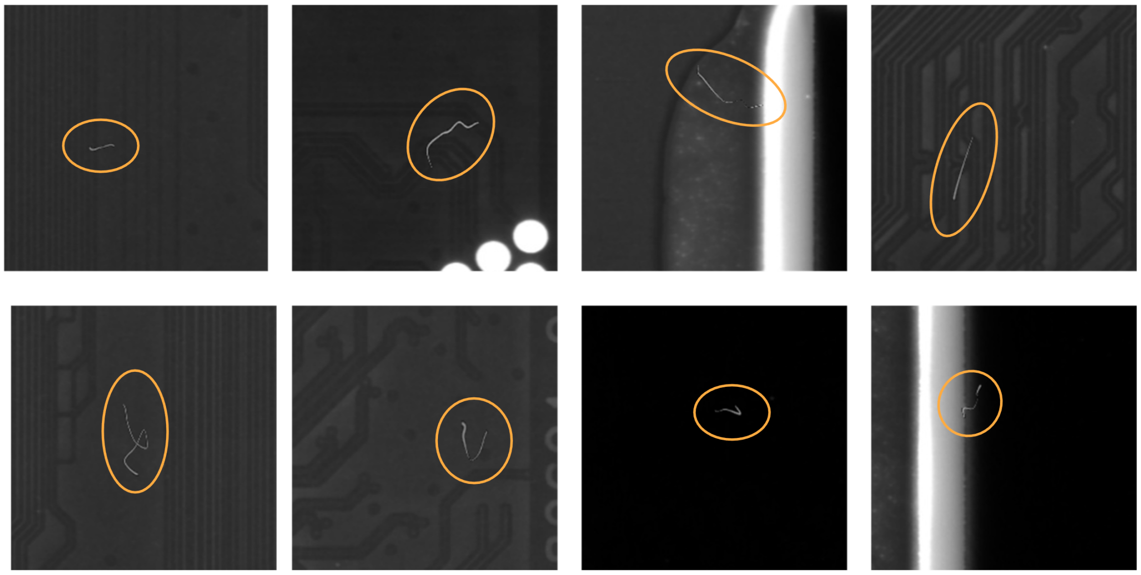


Figure 3.10: Rendered FM Defects Merged with Non-defect Input Images.

containing FM defects and the other half non-defect images randomly cropped from the clean images. We train the classification model with an SGD optimizer and cosine learning rate decay, starting from $1e^{-1}$ to $1e^{-6}$. For the test dataset, we center crop FM defect of size 256×256 from the 47 test images and other 47 non-defect patches cropped randomly from clean images not used in training. We perform two experiments: 1) the model is only trained on the real FM images, and 2) the model is trained on real + rendered images. In addition, we also vary the number of real FM images by randomly selecting fewer real FM images from the training set and plotting their accuracy performance. We perform each experiment three times and report the mean and standard deviation.

Figure 3.11 and Table 3.5 shows the classification accuracy of model trained with different training sets. The first column in the table denotes the number of real FM images used in the training set. The second column shows the accuracy of a model trained only on the real FM defect images. The third column shows the classification performance of the model trained with real + rendered images. When a full real FM

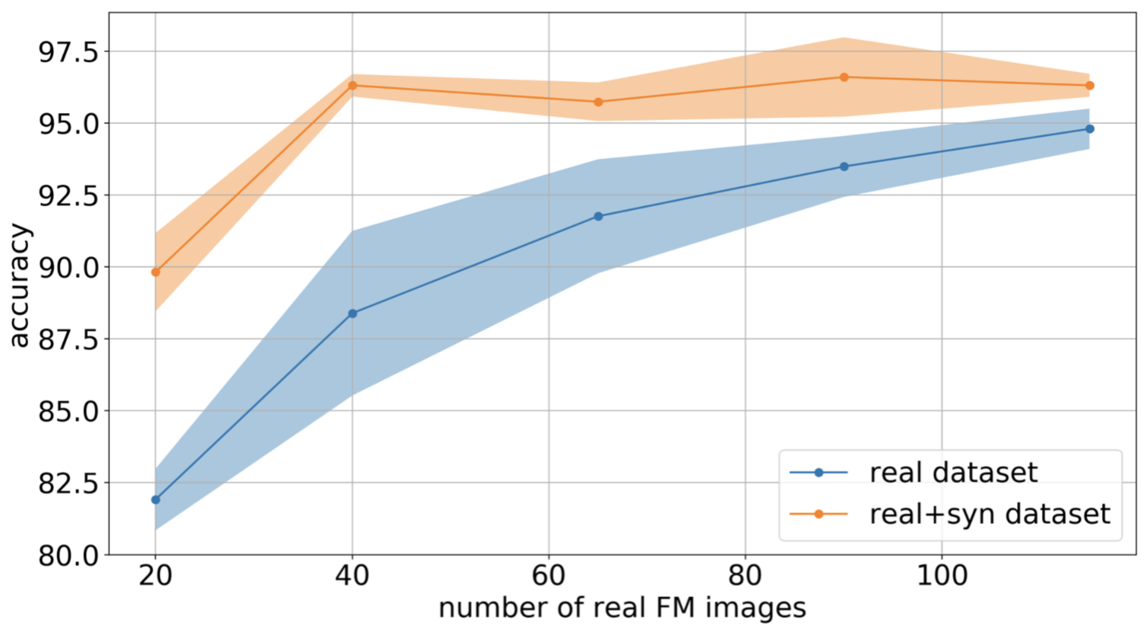


Figure 3.11: Classification Performance on FM Defects with Varying Number of Training Images.

Table 3.5: Performance Comparison of FM Classification Model with Varying Number of Training Images. We Observe That with Fewer Real Training FM Images, the Model Trained Using Real + Rendered Dataset Provides a Gain of $\sim 8\%$ (See 4^{th} Row).

# Real FM images	Accuracy (%)	Accuracy (%)
in training set	Real images	Real + Rendered images
115	94.80±0.70	96.31±0.4
90	93.49±1.06	96.60±1.38
65	91.76±1.98	95.74±0.67
40	**88.39±2.86**	**96.29±0.39**
20	81.92±1.07	89.82±1.36

training set (115 images) is used, we find that the performance improvement for the model trained using real + rendered dataset is roughly $\sim 1.5\%$ over the model trained on only real FM images. As we decrease the number of real FM examples in the training set, we observe a decline in the test performance; however, the model trained on the real + rendered dataset decreases slowly. With fewer real FM images in the training set, the model trained with real + rendered provides a gain of $\sim 8\%$ (see 4^{th} row in Table 3.5). This result demonstrates that the rendered images are very effective when we have fewer real defect images to train the deep learning models.

Epoxy Rendering

We use a CNN-based generator model that takes a binary masked non-defect image as input and generates realistic-looking Epoxy defects on the masked region. Our generator consists of 4 CNN layers, batch normalization, and ReLU layers. To train this generator, we first segment out the real Epoxy defect regions, then construct a polygonal mask around the defect region, and use it as input to the generator as shown in Figure 3.7. The generator is trained to generate the original Epoxy defect back to the masked region. We use the Adam optimizer with a learning rate of $1e^{-4}$ and train it for $30K$ iteration with a batch size of 16 and image size of 256×256. Once trained, we can feed any random polygonal mask and random masked non-defect image to generate realistic-looking Epoxy defects. Figure 3.12 shows an example of generated Epoxy defects from our generator model. Note that we can generate Epoxy defects of any size and location. However, real epoxy defects have inherent location bias, usually found around the die region. Hence while using our model as augmentation, we sample the polygon mask around the die.

We employ the same approach used for FM defects to quantify the quality of generated Epoxy defects. We use ResNet-18 as a binary classification model. We create the training set by randomly extracting the 256×256 patches around the Epoxy defect region from the training set. For synthetic images, we rendered around 500 Epoxy defect images of size 256×256. Test images are created by extracting 50 patches from the 8 Epoxy defect images in the test set and 50 patches from non-defect

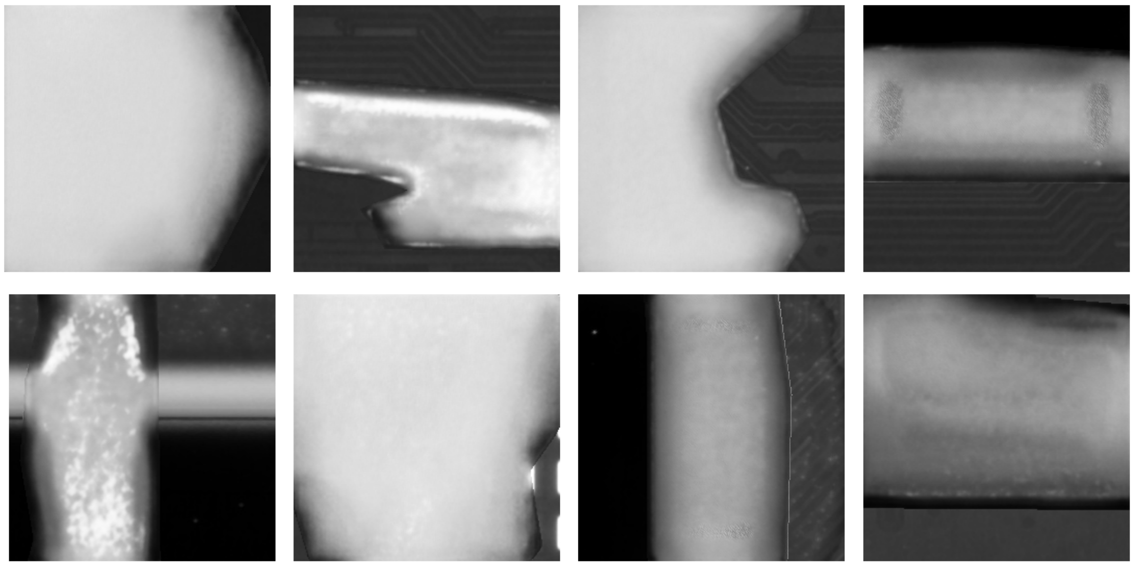


Figure 3.12: Examples of Rendered Epoxy Images from Our Model.

images not used in the training set. We train the generator up to $30K$ iteration with batch size of 32 and cosine learning rate decay from $1e^{-1}$ to $1e^{-6}$.

Table 3.6: Performance Comparison of Epoxy Classification Model with Varying Number of Training Images. The Model Trained Using Real + Rendered Dataset Provides a Gain of $\sim 4\%$ Compared to the Model Trained Only on the Real Epoxy Defects (See 4^{th} Row).

# Real Epoxy images in training set	Accuracy (%) Real images	Accuracy (%) Real + Rendered images
24	95.85±0.32	96.69±1.25
18	94.05±1.40	94.71±0.49
12	93.39±3.21	93.37±2.80
7	**89.24±8.24**	**93.15±3.65**
3	78.55±9.82	78.54±6.40

Table 3.6 show accuracy of ResNet-18 model trained under different datasets. From the table, we observe that when a higher number of real Epoxy defect images are available in training, the model trained using the real+rendered dataset provides a

minor improvement of roughly $\sim 1\%$. However this improvement is much higher when a smaller number of real Epoxy images is available, even showing a nearly 4% points improvement when only 7 real images are available (see 4^{th} row in Table 3.6). Below that level, there seems to be no significant value for adding synthetic defects. This indicates there may be some minimum threshold for the number of real images to see the benefits of dataset augmentation with synthetic data. Compared to the FM classification model, the variance in the performance of the Epoxy classification model is very high when trained with few real Epoxy training images. The high variance in performance is mainly due to the random selection and significantly small number of Epoxy defect images compared to the FM defect images.

3.4.3 Discussion

The proposed hybrid model generates realistic-looking defects of various shapes and sizes while using a small training dataset. Additionally, as a data augmentation technique for the defect classification model, our hybrid model provides a significant performance boost when the number of real defect images are few. These findings are contrary to standard GAN models that require large training datasets in order to generate diverse defects, and need additional human screening to use the synthetic defect images as data augmentation. As we know that modeling all defect types using a rendering tool is not straightforward. However, we believe that our polygon mask-based approach could be extended to 'subtractive' defects like cracks and scratches. In our model we sample shape and size of the rendered defects from real distribution; as random shape and size might not be effective for data augmentation. This requires domain knowledge about the defect's shape and size.

3.5 Conclusion

In this work, we investigate a hybrid approach to enable GAN models to be more effective for data augmentation by incorporating a physics-based renderer. The standard GAN models like style transfer GAN or cycle GAN generate realistic-looking defects and boost the performance of a defect detection model. However, these GAN models require an extensive training dataset to generate all variations of defects, and collecting such large datasets is costly and time-consuming, potentially precluding the use of AI early on in the product/process life-cycle. Furthermore, controlling the shape, size, or other properties of the generated defects from these GAN models is challenging as these variations are often represented in a non-disentangled and non-interpretable manner in the latent space. We propose a hybrid approach combining a 3D rendering tool with a small GAN model to overcome this challenge. We model defects like FM as spline curves or Epoxy defects as polygonal masks with easily controllable parameters. The GAN model in our hybrid approach is only used to refine or apply realistic-looking texture on the rendered defects, in contrast to generating defects from scratch in conventional GAN models. The proposed hybrid model requires a very small training dataset to generate a realistic-looking and diverse defect structure. Thus this approach is more suitable for enabling defect detection and classification in semiconductor manufacturing when few images of the defects are available for training. We evaluate the quality of the synthetic defects by using them as a training augmentation for a defect classification model. We further show that the proposed method significantly improves defect classification performance in small training data scenarios. In future work, we expect to extend our hybrid modeling technique to all possible defect types found throughout the manufacturing line.

Chapter 4

NON-PARAMETRIC PRIOR FOR IMPROVED INTERPOLATION

4.1 Introduction

Advances in deep learning have resulted in state-of-the-art generative models for a wide variety of data generation tasks. Generative methods map sampled points in a low-dimensional latent space with known distribution to points in high-dimensional space with distributions matching real-data. In particular, generative adversarial networks (GANs) [41] have shown successful applications in super-resolution [80], image-to-image translation [61, 175], text-to-image translation [113], image inpainting [106], image manipulation [174], synthetic data generation [130] and domain adaptation [148].

A GAN architecture consists of a generator G and a discriminator D. The generator G maps low-dimensional latent points $z \sim \mathcal{P}_z$ to high-dimensional data distribution $\mathcal{P}_{data}$. The latent-space distribution $\mathcal{P}_z$ is typically chosen to be a normal or uniform distribution. The goal of the generator G is to produce data such that they are perceptually indistinguishable from real data. However, the discriminator D is trained to distinguish between 'fake' and 'real' data. Both the generator and discriminator are trained in an adversarial fashion, and at the end of training the generator learns to generate data with a distribution similar to the real one.

One natural question for generative models is how to model the latent space effectively to generate diverse and varied output. Interpolating between samples in the latent space can lead to semantic interpolation in image space [110]. Interpolation can help transfer certain semantic features of one image to another. Successful interpolation also shows that GANs do not simply over-fit or reproduce the training set, but generate novel output. Interpolation has been shown to disentangle factors of variation in the latent space with many applications [91, 77, 92, 161].

Imposing a parametric structure on the latent space can cause distributional mismatches where the prior distribution does not match the interpolated point's distribution. This mismatch causes the interpolated points to lose fidelity in quality [155]. Previous research has resulted in various parametric models to fix this problem [155, 71, 2]. One of the findings in prior work [82] is that the use of a Cauchy distributed prior solves the distributional mismatch problem. But, Cauchy is a very peculiar distribution, with undefined moments and a heavy-tail. This means that during inference there will always be a number of undesirable outputs (as acknowledged also in [82]) due to latent vectors being sampled from these tails.

In this work, we propose the use of **non-parametric priors** to address the aforementioned issues. The advantage of a non-parametric prior is that we do not use any modeling assumptions and propose a general optimization approach to determine the prior for the task at hand. In particular, our contributions are as follows:

- We analyze the distribution mismatch problem in latent-space interpolation using basic probability tools, and derive a non-parametric approach to search for a prior which can address the distribution mismatch problem.
- We present algorithms to solve for the prior using off-the-shelf optimizers, and show that obtained priors have interesting multi-lobe structures with fast decaying tails, resulting in mid-point distribution to be close to the prior.
- We show that the resulting non-parametric prior yields better quality and diversity in generated output, with no additional training data nor any added architectural complexity.

More broadly, our approach is a general and flexible method to impose other constraints on latent-space statistics. Our goal is not to outperform all the latest developments in generative models, but to show that our proposed stand-alone formulation can boost performance with no added training or architectural modifications.

4.2 Background and Related Work

Generative Adversarial Network: As described in Section 4.1, a GAN consists of two components: a generator G and a discriminator D, which are adversarially trained against one another until the generator can map latent-space points to a high dimensional distribution which the discriminator cannot distinguish from true data samples. Formally, this can be expressed as a min-max game in (4.1) which the generator tries to minimize and the discriminator tries to maximize [41]:

$$\min_G \max_D V(D, G) = E_{x \sim \mathcal{P}_x}[\log(D(x))] + E_{z \sim \mathcal{P}_z}[\log(1 - D(G(z)))], \tag{4.1}$$

where, V is the objective function, $x \sim \mathcal{P}_x$ are real data points sampled from a true distribution, and $z \sim \mathcal{P}_z$ are sampled points from the latent-space distribution. If the training of the GAN is stable and the Nash equilibrium is achieved, then the generator learns to generate samples similar to the true distribution. In general, GAN training is not always stable, thus several methods have been introduced to improve the training [122, 7]. This includes different kinds of divergences and loss functions [103, 8, 44]. Several other works improve generated image quality [21, 169] or resolution [24, 66].

Interpolation: For any two given latent-space points $z_1, z_2 \sim \mathcal{P}_z$, a linearly-interpolated point z_λ is given by $z_\lambda = (1 - \lambda)z_1 + \lambda z_2$ for some $\lambda \in [0, 1]$. It has been shown that GANs can generate novel outputs via linear interpolation, and as the line is traversed ($\lambda : 0 \to 1$), the output images smoothly transition from one to another without visual artifacts [110]. They further showed that vector arithmetic in the latent space has corresponding semantic meaning in the output space, e.g. latent-space points for “man with glasses” - “man without glasses” + “woman without glasses” generates an image of a woman wearing glasses (c.f. Fig. 7 of [110]).

Distribution Mismatch of Interpolated Points: Interpolation, while semantically meaningful, presents challenges in ensuring all interpolated points preserve the same data quality (or in the case of images visual quality). Most GANs utilize simple parametric distributions such as normal or uniform as the prior distribution to sample the latent space. However, these two choices of priors cause the interpolated point's distribution to not match with either the normal or uniform distribution as observed by [71]. We replicate this argument below for the sake of exposition, since this is the core problem we tackle in this work.

Let $z_1, z_2 \sim \mathcal{N}(0, \sigma^2 I)$ be two points in the latent space of the GAN's generator, and let $z_\lambda = (1-\lambda)z_1 + \lambda z_2$ be a linearly interpolated point. Note that $\sigma^2_{z_\lambda} = (1-\lambda)^2\sigma^2_z + \lambda^2\sigma^2_z$. We are interested when $\sigma^2_{z_\lambda} = \sigma^2_z$, which only holds for delta functions for finite moment distributions (we later prove this statement more formally in Section 4.3). However, we see that the worst case for when $\sigma^2_{z_\lambda}$ is different from σ^2_z occurs at $\lambda = 1/2$ or the midpoint denoted $z' = \frac{z_1+z_2}{2}$. Analyzing the Euclidean norm of z_1, z_2 and z' gives the following equations [71]:

$$\begin{aligned} z_1, z_2 \sim \mathcal{N}(0, \sigma^2 I) \Rightarrow &\quad \|z_1\|^2, \|z_2\|^2 \sim \sigma^2 \mathcal{X}^2(d), \\ z' = \frac{z_1 + z_2}{2} \Rightarrow &\quad \|z'\|^2 \sim \frac{\sigma^2}{2}\mathcal{X}^2(d), \end{aligned} \tag{4.2}$$

where, d is the dimension of the latent-space and $\mathcal{X}^2$ is the chi-squared distribution. The GAN is trained with latent-space whose norm squared distance follows a $\sigma^2\mathcal{X}^2(d)$ distribution according to (4.2). However, the mid-point will have a distribution $\frac{\sigma^2}{2}\mathcal{X}^2(d)$. Clearly, there is distribution mismatch between the points at which the GAN is trained to generate realistic samples and the mid-point where we want to do interpolation. Further, this distribution mismatch becomes worse if we increase the dimension of the latent space. Finally, it has been shown that the Normal distribution in higher dimensions forms a 'soap bubble' structure, i.e. most of its probability mass concentrates in an annulus around the mean, rather than near the mean itself [46]. This implies that interpolations that traverse near the origin of the latent space will suffer degradation in output fidelity/quality, which has been confirmed for GANs in [155]. A similar proof for a distribution mismatch can be shown for the uniform distribution.

4.3 Design of Non-parametric Priors for GANs

The primary motivation to design non-parametric priors is that in order to have a prior whose distribution of mid-points is close to the original prior, we need to optimize for an appropriate cost over the space of density functions. This optimization is easily done for the non-parametric case, and requires rather few assumptions. Our terminology of *non-parametric* stems from classical density estimation approaches, rather than the more modern usage in Bayesian non-parametric.

Let $f_X(x)$ be the chosen prior distribution. Let x_1 and x_2 be two samples drawn from $f_X(x)$. Let an interpolated point be given by: $(1-\lambda)x_1 + \lambda x_2$, for $0 < \lambda < 1$. The precise relation between the distribution of this interpolated point and $f_X(x)$ is given analytically as follows.

Property 1: If x_1 and x_2 are two independent samples drawn from $f_X(x)$, then the density function of $(1-\lambda)x_1 + \lambda x_2$, for $0 < \lambda < 1$ is given by $\frac{1}{|\lambda(1-\lambda)|} f_X(\frac{x}{\lambda}) * f_X(\frac{x}{1-\lambda})$.

Proof: The proof is a direct application of the following two results from probability theory.

- R_1: If X_1 and X_2 are two independent random variables, with density functions

$f_{X_1}(x)$ and $f_{X_2}(x)$, then the density of their sum $X_1 + X_2$ is given by the linear convolution $f_{X_1}(x) * f_{X_2}(x)$.

- R_2: If random variable X has density function $f_X(x)$, then for $\alpha \in \mathbb{R}$, the density of αX is given by $\frac{1}{|\alpha|} f_X(\frac{x}{\alpha})$.

Apply R_2 to $(1-\lambda)x_1$ and λx_2 separately, then convolve the results using R_1. QED.

Following from here, the distribution mismatch problem can be expressed as the search for a prior distribution $f_X(x)$ such that the distribution of any other interpolated point is close to $f_X(x)$. That is, we would like $f_X(x)$ to satisfy:

$$f_X(x) = \frac{1}{|\lambda(1-\lambda)|} f_X(\frac{x}{\lambda}) * f_X(\frac{x}{1-\lambda}), \tag{4.3}$$

where, $\lambda \in (0, 1)$. The only distributions we are aware of that satisfy this condition are the Cauchy (undefined moments, heavy-tailed), and delta functions (zero variance).

Property 2: The only density functions with finite moments that satisfy condition (4.3) are delta functions.

Proof: A density function that satisfies condition (4.3), will also satisfy the equality of all moments of the left and right side densities. By specifically applying this to the equality of variances, we will show that the only solution is a delta function (among the class of finite moment densities). The following two results come handy.

- R_3: If X_1 and X_2 are two independent random variables, with respective variance σ_1^2 and σ_2^2, then the variance of their sum $X_1 + X_2$ is given by $\sigma_1^2 + \sigma_2^2$.
- R_4: If random variable X has variance σ^2, then for $\alpha \in \mathbb{R}$, the variance of αX is given by $\alpha^2\sigma^2$.

If (4.3) holds, it must imply the equality of the variance of the prior, and the variance of any intermediate-point. i.e. $\sigma_X^2 = (1-\lambda)^2\sigma_X^2 + \lambda^2\sigma_X^2$. If σ_X is finite, equality happens if and only if $\sigma_X = 0$. This implies that $f_X(x)$ is a delta function. QED.

The search for a density function that satisfies (4.3) is thus not meaningful in the context of generative models, because a delta function as prior implies constant output. Cauchy, on the other hand, is too specific a choice, and suffers from pathologies such as undefined moments, which renders imposing any additional constraints on latent-space statistics impossible. It also is heavy-tailed, which may cause generation of undesirable outputs for samples from the tails.

What if we relax condition (4.3), such that we do not seek exact equality, but closeness of the left and right sides? The next section shows that this relaxed search results in a problem which can be solved using standard off-the-shelf function minimizers. Using this approach we obtain distributions with many useful properties. If we let $P(x) = f_X(x)$, and $Q(x;\lambda) = \frac{1}{\lambda(1-\lambda)} f_X(\frac{x}{\lambda}) * f_X(\frac{x}{1-\lambda})$, we seek to minimize some form of distance or divergence between $P(x)$ and $Q(x)$ among densities with finite-variance. This optimization problem is defined in the next section.

4.3.1 Searching for the Optimal Prior Distribution

As mentioned earlier, instead of enforcing exact equality as in condition (4.3), we would like to minimize the discrepancy between the left and right sides. A natural choice would be to minimize the KL divergence between $P(x) = f_X(x)$, and $Q(x;\lambda) = \frac{1}{\lambda(1-\lambda)} f_X(\frac{x}{\lambda}) * f_X(\frac{x}{1-\lambda})$. Ideally, it might make sense to minimize this over the entire range of λ's, i.e.

$$\underset{P}{\text{minimize}} \int f(P(x) \| Q(x;\lambda)) d\lambda, \tag{4.4}$$

where f is the chosen divergence/distance between the densities. However, this is likely an intractable problem due to integration over λ. In order to make this tractable, we observe that for a given λ, the mean of $Q(x;\lambda)$ is the same as the mean of $P(x)$. However, the variance of $Q(x;\lambda)$ goes as $(1-\lambda)^2\sigma_X^2 + \lambda^2\sigma_X^2$. Thus, for interpolation problems, where $\lambda \in (0,1)$, the largest discrepancy in variance between P and Q occurs at a value of $\lambda = 0.5$. Thus, for interpolation problems, we suggest that minimizing for the worst-case error is sufficient.

4.3.2 Optimization Problem for Interpolation Priors

For interpolation priors, we minimize the KL divergence between $P(x)$ and $Q(x;\lambda = 0.5)$. To create a tractable problem, we restrict $P(x)$ to be defined over a compact domain, without loss of generality, we choose it to be $[0,1]$. We discretize the domain with sufficient fineness, typically we choose 2^{10} bins in $[0,1]$. The distribution is now discretely represented by the bin-centers $P = \{p_i\}_{i=1}^n$. The optimization problem now becomes:

$$\min_P f(P\|Q) \text{ s. t. } \sum_{i=1}^n p_i = 1, \text{ and } p_i \geq 0. \tag{4.5}$$

In (4.5), $f(P\|Q)$ is a divergence/distance function between P and Q. We use KL divergence because not only is it a natural choice, but we also find that it produces smooth distributions than when using the ℓ_2 distance. Without a variance constraint, the solution of (4.5) is simply a discrete delta function, which we would like to avoid. The variance constraint is equivalently expressed as a quadratic-term involving p_i's, based on which we have:

$$\begin{aligned} &\min_P f(P\|Q) \\ \text{s.t.} \sum_{i=1}^n p_i = 1,\ &\frac{1}{n}\left(\sum_{i=1}^n i^2 p_i - \left(\sum_{i=1}^n i p_i\right)^2\right) \geq \xi, p_i \geq 0, \end{aligned} \tag{4.6}$$

where, $\xi > 0$. In general the KL divergence is convex on the space of density functions, but in our case P and Q are related to each other; our objective function is not convex. We solve (4.6) using *fmincon* in Matlab, which uses an interior-point algorithm with barrier functions. We note that the solution from fmincon may only be a locally optimal solution, yet we find the obtained solution is quite robust to large variation in initialization. We also note that using any of $KL(P\|Q)$, $KL(Q\|P)$ and $KL(P\|\frac{(P+Q)}{2}) + KL(Q\|\frac{(P+Q)}{2})$ as objective in (4.6) gives us the same result. We use the following settings for *fmincon*: interior-point as algorithm, *Max Function Evaluations* $= 4 \times 10^5$, *Max Iterations* $= 10^5$ and $n = 1024$. We use $\xi = 0.75$ in our experiments because it provides the best FID score [51]. Figure 4.1 shows the trace of the optimizer cost value for the above settings; we observe convergence to a local minimum in less than 500 iterations.

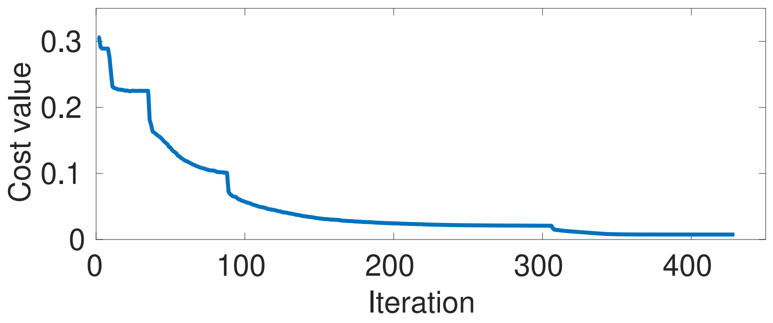


Figure 4.1: Figure Shows a Sample Trace of the Cost Function (4.6) over Iterations, Showing Fast Convergence.

Remarks on the shape of the obtained distribution: Here we make brief remarks on the shape of the obtained distribution. Firstly, we note that the exact shape of the obtained distribution varies slightly each time we run the solver. This is of course expected. However, we find that all obtained solutions seem to share the same general shape: they have a large main-lobe, appear to be symmetric, and have small but significant side-lobes. This is more clearly shown in Figure 4.2. We

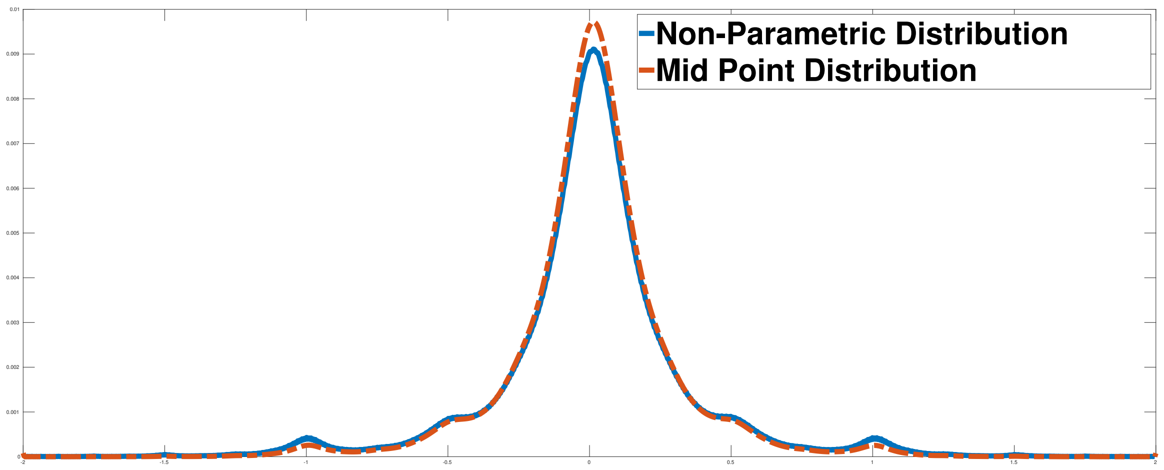


Figure 4.2: The Distribution Obtained by Solving (4.6) (Shown in Blue), and Its Mid-point Distribution (Red). While There Is Small Variability in the Solutions Obtained, We Find That No Matter How We Initialize the Solver, All Our Obtained Distributions Share the Three Following Traits: A Large Main-lobe, Symmetry, and Small Side-lobes. Also, Note the Strong Overlap Between the Distribution and the Mid-point Distribution. This Is Further Quantified in Table 4.1 and Compared with Other Distributions in Figure 4.4.

note that we did not impose any symmetry condition during optimization, yet these solutions emerged despite different initialization.

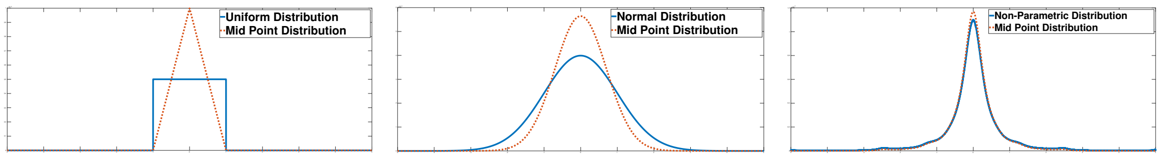
Dependence on initialization: We initialized our solver with a uniform density, delta functions centered at different locations, and truncated Gaussians with varying means and standard-deviations. For all these, the final solution still converges to a shape very similar to that shown in Figure 4.2. Further, all obtained solutions seem to perform equally well in the final evaluation of GAN output quality.

Role of side-lobes: We are not aware of any simple parametric distribution that can describe the shape seen in Figure 4.2, except perhaps a Gaussian mixture model. However, the shape of the side-lobes is intricate, and not simple Gaussian-like. The existence of these side-lobes seems to allow us to strike a balance between the fast tail-decay of distributions like the Gamma, and the heavy tail of the Cauchy. It is almost as if the obtained shape fuses the best properties of the two classes of distributions, enabling us to generate good quality GAN output, all the while minimizing the divergence to the interpolated samples.

Continuous samples from discretized density: At first glance it may appear that since we discretize the domain $[0, 1]$ while solving (4.6), that our prior is capable of generating only discrete samples. This is easily dealt with as follows. Once we generate a sample from the discretized density, what we get is really an index corresponding to the bin-center, but the bin itself has non-zero width given by how finely we partition $[0, 1]$. From the corresponding bin, we simply generate a uniform random variable restricted to the width of the bin. This approach implicitly corresponds to sampling from a continuous density constructed by a zeroth-order interpolation over the obtained discrete one. One can be more sophisticated than this, but the sampling algorithm will no longer be as simple. We find that the approach described above is quite sufficient in practice.

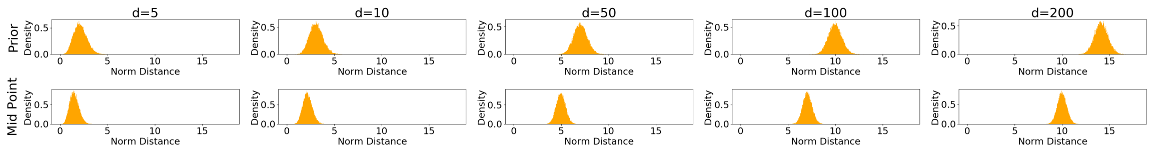
Quantification of mid-point mismatch: Table 4.1 shows the actual KL divergence between the prior and mid-point distribution. It is clear that for the normal and uniform distributions, the mid-point distribution is very divergent from the actual prior distribution; whereas the distribution obtained from solving (4.6) has a much lower divergence from the mid-point. In (4.6) we are only minimizing the distribution mismatch for the one-dimensional case. The idea is that if the 1-D distribution is similar to its mid-point distribution, then the divergence between the corresponding Euclidean norm distribution will be low even for higher dimensions.

Figure 4.3 shows the mid-point distribution mismatch for different priors for the one dimensional case. Figure 4.4 shows the Euclidean norm distribution for prior and mid-points for different dimensions, computed from a set of 5×10^4 samples. For

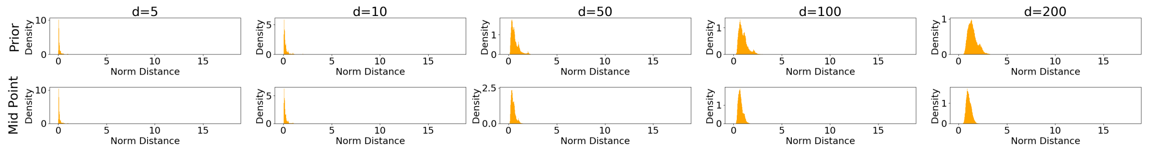


(a) Uniform Distribution (b) Normal Distribution (c) Non-parametric

Figure 4.3: The Figures Show Various Choices of Priors (Blue) and Their Corresponding Mid-point Distribution (Red). Note That One Can Observe a Large Discrepancy Between the Prior and Mid-point Distributions, for Typical Choices Such as the Uniform and Normal. The Prior We Develop Shows Significantly Less Discrepancy. These Discrepancies Are Also Quantified via the Kl-divergence in Table 4.1.



(a) Normal Distribution



(b) Obtained Non-parametric Distribution

Figure 4.4: Euclidean Norm Distribution for Samples Drawn from Different Priors and Their Corresponding Mid-point Euclidean Norm Distribution for Different Dimensions d. Note the Mid-point Norm Distribution for the Normal Prior Moves Further Away from the Prior Norm Distribution as the Dimension Increases to $d = 200$, Whereas with Our Non-parametric Prior, the Mid-point Norm Distribution Overlaps with the Prior Norm Distribution Even at $d = 200$.

the mid-points, we sample two sets of 5×10^4 points, and calculate the Euclidean norm of the corresponding mid-points. We see that for the normal distributions, at low dimensions ($d = 5$ and $d = 10$) the mid-point distribution overlaps well with the prior distribution. As the dimension increases ($d = 50, 100, 200$), the two distributions start to diverge. For $d = 100, 200$ there is almost no overlap between the prior and mid-point distribution. We observe similar trend for the uniform distribution. On the contrary, in our case the mid-point distribution and the prior distribution (of Euclidean norm) overlap well with each other even in higher dimensions. In Figure

Table 4.1: KL-divergence Between Prior and Mid-point Distribution.

Distribution	KL divergence
Uniform Distribution	0.3065
Normal Distribution	0.1544
Proposed Non-Parametric Distribution	**0.0075**

4.4 we can notice that our non-parametric distribution does bring the Euclidean norm distribution very close to the origin compared to the normal and uniform.

We note that we are not the first to propose a solution to this problem. Several prior approaches have proposed solutions, either through new interpolation schemes or new prior distributions different from normal or uniform that suffer less distribution mismatch. White et al. proposes spherical linear interpolation by following the geodesic curve on a hypersphere to avoid interpolating near the origin (and thereby minimize distribution mismatch) if the latent points are sampled uniformly on a sphere with finite radius [155]. However, this interpolation is not semantically meaningful if the path becomes too long and it passes through unnecessary images as noted in [71]. Similar to spherical interpolation, [2] propose a normalized interpolation. Yet, it inherits similar issues as in spherical interpolation in not being the shortest path.

Other approaches have attempted to define new prior distributions to ensure the interpolated points have low distribution mismatch. This is similar to the method we employ in our work, except ours is non-parametric. Kilcher et al. propose a new prior distribution defined as follows:

$$v \sim Uniform(S^{d-1}), r \sim \Gamma(\frac{1}{2}, \theta), z = \sqrt{r}v \tag{4.7}$$

where d is the dimension of the latent space, $\Gamma(\frac{1}{2}, \theta)$ is the Gamma distribution and z is a latent vector [71]. Latent spaces defined using this prior distribution do not suffer as much from mid-point distribution mismatch. Further work by Lesniak et al. showed that the Cauchy distribution $P(z) = \frac{1}{\pi(1+z^2)}$ induces a midpoint distribution which is the same as itself [82]. However, the Cauchy distribution has undefined moments, which makes analysis difficult, and also is heavy-tailed, which can lead to undesirable outputs.

4.4 Experiments and Results

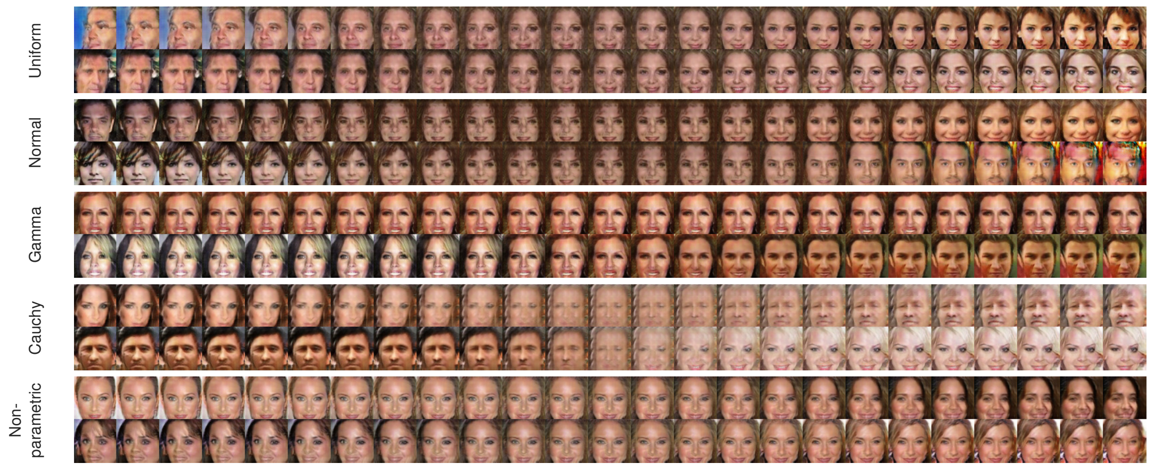


Figure 4.5: Interpolation (Left to Right) Through the Origin on Celeba Dataset Using Different Priors with $d = 100$. Note the Degradation in Image Quality Around the Center of the Panel (Origin Space) for Many Standard Priors.

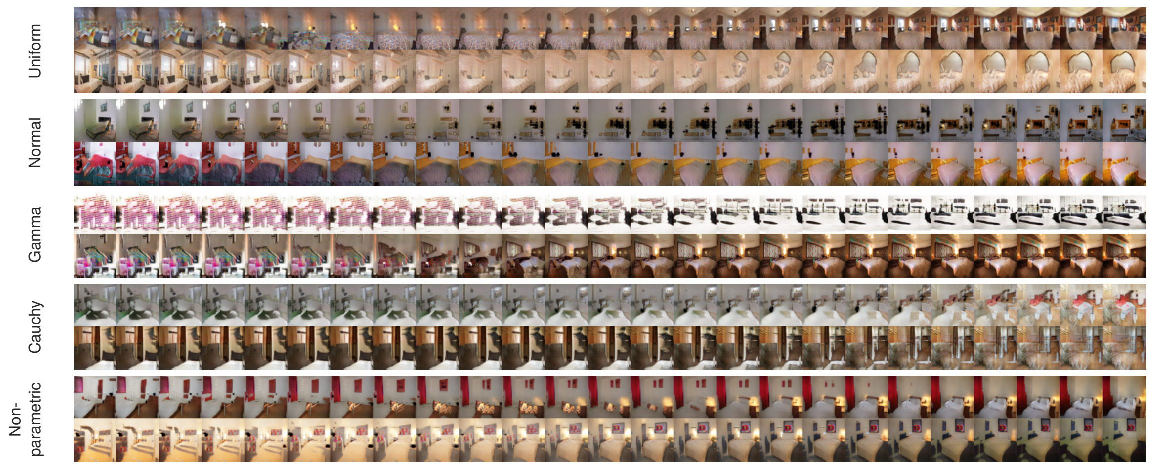


Figure 4.6: Interpolation (Left to Right) Between Two Random Points on Lsun Bedroom Dataset Using Different Priors with $d = 100$.

Datasets, models, and baselines: To validate the effectiveness of the proposed approach, we train the standard DCGAN model [110] on four different datasets: a) CelebA dataset [93], b) CIFAR10 [75], c) LSUN Bedroom, and d) LSUN Kitchen [164]). We train our model to the same number of epochs and all the hyper-parameters of the training are kept same for all the cases. We train each model three times and report the average scores. Details about the network architecture and the training method are provided in the supplemental material. We compare our proposed non-parametric prior distribution against standard ones like the normal, uniform and the priors designed to minimize the mid-point distribution mismatch like Gamma [71], and Cauchy [82]. For Gamma and Cauchy, we use the same parameters as suggested in the corresponding references.

Qualitative tests: In Figure 4.5, we show the effect of interpolation through origin in high latent-space dimension ($d = 100$) for different priors on CelebA datatset. Here, we interpolate between two random points such that the interpolation line passes through the origin. Similar to [71], we also observe that with standard priors like the normal, in high latent dimension, the GAN generates non-realistic images around the origin. Note the difference in quality near the images in the center of the panels (space around origin). With our non-parametric distribution obtained from the solution of (4.6), the GAN generates more realistic images around the origin even at higher dimensions. It was pointed out by Lesniak et. al. [82] that if a GAN is trained for more epochs, then it learns to fill the space around the origin even with the standard priors. We observe similar trend with the normal and uniform priors. However, with our non-parametric prior the GAN learns to fill the space around the origin very early in training compared with the standard priors. We also present qualitative comparisons on LSUN bedroom dataset in Figure 4.6: comparing the results with the standard priors and the priors proposed in [71] and [82], highlighting the favorable performance of the proposed approach. While we note that it is difficult to perceptually appreciate whether we outperform the other priors, we show that we do obtain competitive visual quality with a conceptually general approach. In the supplemental material, we show additional qualitative results for LSUN Bedroom/Kitchen and CIFAR10 datasets. We note that the Cauchy distribution had difficulty converging on these datasets, exhibiting possible mode collapse and instability during GAN training.

Quantitative evaluation: For quantitative analysis, we use the Inception Score (IS) [122] and the Frechet Inception Distance (FID) [51] which are the standard metrics used to evaluate GAN performance. The inception score *correlates* with the visual quality of the generated image – higher the better. However, recent studies suggest that the inception score does not compare the statistics of the generated dataset with the real-world data [51, 166], and thus is not always a reliable indicator of visual quality. This drawback of the IS is overcome by the FID score, which compares the statistics of the generated data with the real data with respect to features. The lower the FID score, the better. We will see in Tables 4.2 to 4.6, that our non-parametric prior performs better in terms of FID score on both the prior and mid-point by at least 2 points. In terms of IS we are the best in most of the cases, when we do not perform better, we come quite close to the best performing one.

To get the IS and FID score we sample 5×10^4 points from the prior, and estimate the scores on the corresponding image samples. For mid-point, we sample two sets of 5×10^4 points from the prior, and an image is generated by the GAN, with the corresponding average points as inputs. Results are summarized in Tables 4.2 to 4.6 for different datasets. Table 4.2 compares our non-parametric prior with other standard priors on the CelebA dataset, at latent space dimension $d = 100$. The non-parametric

Table 4.2: Comparison of IS and FID Scores for Different Prior Distributions on CelebA Dataset with $d = 100$.

Distribution	Inception Score		FID Score	
	Prior	Mid-Point	Prior	Mid-Point
Uniform	1.843	1.369	24.055	40.371
Normal	1.805	1.371	26.173	42.136
Gamma	1.776	1.618	29.912	28.608
Cauchy	1.625	1.628	59.601	60.128
Non-parametric	**1.933**	**1.681**	**17.735**	**19.115**

prior outperforms all other priors on both the metrics. Our prior has better FID score by more than 6 points on both the prior and the mid-point. As expected the IS and FID scores for the Cauchy is almost the same for the prior and the mid-point. With Gamma, we observe that the score on mid-points is slightly better than the prior since the gamma distribution is highly dense around the origin. In Table 4.3, we show the IS and FID for the prior and the mid-point at latent space dimension $d = 200$. We note that our non-parametric distribution performs better in all the cases compared with all other priors. Scores for our non-parametric prior is almost similar to the scores in Table 4.2, which indicates its robustness toward the increase in latent space dimension. Cauchy prior sometimes leads to mode collapse during the training which is indicated by its poor FID scores. From Table 4.2 and 4.3, we also note that the IS and FID score become worse for the mid-point compared to the prior point as we increase the latent space dimension. Table 4.4 shows the IS and FID scores for CIFAR10. With our non-parametric prior, the GAN performs better than other priors on both the prior and mid-point by at least 2 points on FID score. Similar to CelebA dataset, we observe the training with Cauchy prior is highly unstable. In Table 4.5 and 4.6 we compare our non-parametric prior with other priors on the LSUN bedroom

Table 4.3: Comparison of IS and FID Scores for Different Prior Distributions on CelebA Dataset with $d = 200$.

Distribution	Inception Score		FID Score	
	Prior	Mid-Point	Prior	Mid-Point
Uniform	1.908	1.407	25.586	44.837
Normal	1.857	1.434	25.035	43.596
Gamma	1.738	1.608	33.816	32.241
Cauchy	1.734	**1.743**	86.286	86.278
Non-parametric	**1.973**	1.636	**14.953**	**19.322**

Table 4.4: Comparison of IS and FID Scores for Different Prior Distributions on CIFAR10 Dataset with $d = 100$.

Distribution	Inception Score		FID Score	
	Prior	Mid-Point	Prior	Mid-Point
Uniform	6.411	5.204	43.501	76.913
Normal	6.836	5.656	39.235	65.525
Gamma	6.449	6.798	48.334	39.262
Cauchy	2.972	2.964	180.37	180.40
Non-parametric	**6.871**	**6.809**	**34.803**	**37.112**

and LSUN kitchen datasets. We observe that our non-parametric prior outperform other priors on the FID score by at least 6 points. We observe that the Gamma and Cauchy priors perform worse on both the prior and mid-point in terms of FID score. These priors often lead to mode collapse during the training. Note that the LSUN dataset has a larger variation in images, and also a larger training-set than the CelebA dataset. The non-parametric prior performs best in both cases as measured by the FID on both prior and mid-point, showing its benefits on large datasets with large variation. **A few salient observations** from the results are:

- The quantitative results on different datasets show that standard priors like

Table 4.5: Comparison of IS and FID Scores with Different Prior Distributions on LSUN Bedroom Dataset with $d = 100$.

Distribution	Inception Score		FID Score	
	Prior	Mid-Point	Prior	Mid-Point
Uniform	2.969	2.649	42.998	76.412
Normal	2.812	2.591	64.682	108.49
Gamma	2.930	2.808	162.44	161.37
Cauchy	**3.148**	**3.149**	97.057	97.109
Non-parametric	3.028	2.769	**27.857**	**31.472**

Table 4.6: Comparison of IS and FID Scores with Different Prior Distributions on LSUN Kitchen Dataset with $d = 100$.

Distribution	Inception Score		FID Score	
	Prior	Mid-Point	Prior	Mid-Point
Uniform	2.656	2.549	40.041	51.119
Normal	2.844	2.867	39.909	53.448
Gamma	2.183	2.147	181.81	187.00
Cauchy	1.182	1.179	242.27	242.87
Non-parametric	**3.109**	**3.031**	**33.194**	**35.074**

the normal and uniform perform better on the prior point but worse on the mid-point.

- The priors proposed to minimize the mid-point distribution mismatch in [71] and [82] achieve better results on the mid-point but perform worse on the prior point.
- Gamma and Cauchy do not perform consistently across datasets. In some cases they are the best, but when they are not, their performance can be far from the best.
- The non-parametric distribution is far more consistent, and is either the best, or

pretty close to the best performing one, on all four datasets.

4.5 Conclusions

In this work, we propose a generalized approach to solve distribution mismatch for interpolation in GANs. We show the qualitative and quantitative effectiveness of our approach over the standard priors. We note that often times, our proposed method is in fact the best one, and in cases when it is not, it comes quite close to the best performing technique. Our goal is not necessarily outperform all other GANs, but to suggest the use of non-trivial priors, which might improve image quality without any additional training-data or architectural complexity. Additionally, it would be an interesting avenue of future work to extend this approach to extrapolation problems, or impose other interesting statistical or physically-motivated constraints over latent spaces.

Chapter 5

IMPROVING SHAPE AWARENESS AND INTERPRETABILITY IN DEEP NETWORKS USING GEOMETRIC MOMENTS

5.1 Introduction

Advances in deep learning have resulted in state-of-the-art performance for a wide variety of computer vision tasks. The large quantity of training data and high computation resources have made convolutional neural networks (CNN) into a common backbone model for many tasks; including image classification [76, 139, 49], object detection [40, 115, 48], segmentation [117, 14, 13], unsupervised learning [149, 16], and generative modeling [74, 41, 132].

A CNN consists of multiple spatially compact filters which convolve over an input image, followed typically by normalizations [60] and nonlinearities. The convolutional kernel's small spatial extent and weight sharing properties make them efficient and translation equivariant. However, this also implies that the kernel's receptive field is limited due to its small spatial extent. The local nature of the convolution kernels prevents them from capturing the global context of the image. The long-range dependency, i.e., a larger receptive field, is achieved through stacking multiple CNN layers and reducing the spatial dimension by pooling operations.

However, it has been observed that the features from this kind of architecture tend to be more receptive toward texture than the shape of the object. For example, [39] tackles this problem using a better shape-biased dataset like Stylized-ImageNet. As opposed to this, incorporating the shape bias more directly without changing training sets is a natural choice. As we know, convolutional operations intrinsically represent frequency selective operations, while the shape is related to geometric concepts rather than specific frequency bands. Therefore, different types of operations that are more directly shape-sensitive are needed to promote shape awareness.

In this work, we frame vision tasks like classification through the geometrical properties of the object's shape. Rigid and non-rigid aspects of shape can be described in terms of geometric moments. Geometric moments are a very specific type of weighted averages of image pixel intensities, where the weights are drawn from specific polynomial-type basis functions. This operation can be expressed as a projection of the image on the bases. While classic theoretical development around moments used specific choices of the bases functions, their application to difficult tasks like image classification has remained very limited. In this work, we revisit moments as a learnable spatial operation, introducing modules motivated by the image-projection analogy. However, we leave the bases to be learned end-to-end in a task-specific manner.

Geometric moments have a long history in the vision community for a wide variety of applications ranging from invariant pattern recognition [57, 70, 3, 32], segmentation [114, 35] and 3D shape recognition [147, 119, 28]. We propose a deep-learning based architecture to extract invariant image moments for classification tasks. Our architecture consists of two streams of convolutional networks; one extracts features corresponding to the object, i.e., removes the background from the image, and the second network learns the bases from a 2D coordinate grid. Geometric moments are computed by projecting features of the image to the learned bases. In order to learn task-specific invariant moments to deformations, size, and location, we learn a simple transformation of the coordinate grid and compute the geometric moments at multiple levels. The geometric moment captures long-range dependency without using any pooling layer or reducing the spatial dimension. The computation cost of the geometric moment is linear in the spatial dimension.

In particular, the proposed Deep Geometric Moment (DGM) architecture provides four key benefits compared to existing models.

- First, the model generates discriminative features for the classification task by accounting for shape information through the proposed deep geometric moments.
- Second, our model outperforms existing ResNet models on standard datasets without using any pooling layer or reducing the spatial dimension.
- Third, it provides easy access to interpretable features at any level by simple re-projection of moments.
- Finally, compared to existing models, the DGM model only requires finetuning the coordinate basis pipeline without retraining all the model parameters.

Our goal is not to outperform all the latest developments in vision, but to show that our proposed model can perform comparably to standard models when trained from scratch, produces interpretable results, and is easier to finetune.

5.2 Geometric Moment

A *moment* for a given two-dimensional piece-wise continuous function $f(x, y)$ is defined as:

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y) dx dy, \tag{5.1}$$

where, (x, y) is the 2D coordinate and $(p+q)$ is the order of the moment. By uniqueness theorem [57], if $f(x, y)$ is a piece-wise bounded continuous function (i.e. it is non-zero only on a compact part of the xy plane), then the moment sequence m_{pq} is uniquely defined for all orders $(p + q)$ by $f(x, y)$. Conversely, $f(x, y)$ is uniquely determined by the sequence m_{pq}.

Equivalently, moments can also be seen as a 'projection' of the 2D function on certain bases of the form $x^p y^q$. Instead of using the bases function of type $x^p y^q$, one can instead also use orthogonal functions like Legendre or Zernike polynomials [143] for better reconstruction.

Image moments are well-known invariant shape descriptors with a long history of use in the computer vision literature to capture the geometrical properties of an image. For example, m_{00} (0^{th} order) represents average pixel intensity, m_{10} (1^{st} order) and m_{01} (1^{st} order) represent xy centroid coordinate and the combination of 1^{st} and 2^{nd} order can be used to compute orientation.

We need *discriminative moments*, which are also invariant to certain image transformations like rotation, translation, and scale for the image classification task. An early work by Hu [57] introduced a way to find invariant moments for images. The Hu moments consist of seven moments, mostly a combination of lower-order moments invariant under scaling, translation, and rotation. While these basic sets of seven Hu moments are provably invariant to rotation, translation, and scale, their use has been limited since their discriminative power is not very high. Developing invariant moments for the Legendre and Zernike polynomials for any arbitrary order is also possible [19, 167, 168, 70, 72, 153, 159, 31]. However, they also have not significantly impacted contemporary image classification tasks.

In this work, we seek to advance a new approach for defining spatial operations for image classification networks, whose structure is motivated by classic moment computation but whose basis functions are left to be learned end-to-end by a deep learning network in a task-specific way. This implies that we are not seeking to replicate any of the classical moments in an exact sense but to find ways to fuse moment-like computations and let networks learn the suitable basis functions for a given task. This approach is described in the next section.

Geometric moments and deep networks: There has been prior work in integrating geometric moments with deep networks, as specifically applied to 3D shape classification, from point-cloud data. For example, geodesic moment-based features from an auto-encoder were used to classify 3D shapes [94]. On the other hand, CNNs were used as a polynomial function to learn bases, and the needed affine transformation parameters for 3D point cloud data based shape classification [65]. This line of work was extended in [85] which uses graph CNN to capture local features of the 3D object. More recently, [144] and [156] replaced the conventional global average pooling in CNN models with invariant Zernike moment-based pooling for image classification task. Contrary to these methods, our approach learns bases as well as the affine parameters. Note that our work is different from these approaches because a) we are interested in natural image classification where moment computation is challenging compared to 3D shape classification, due to intensity variation, background variation, occlusions, etc, b) architecturally our approach is more involved compared to processing 3D object that are specified directly in terms of coordinate locations.

5.3 Deep Geometric Moments

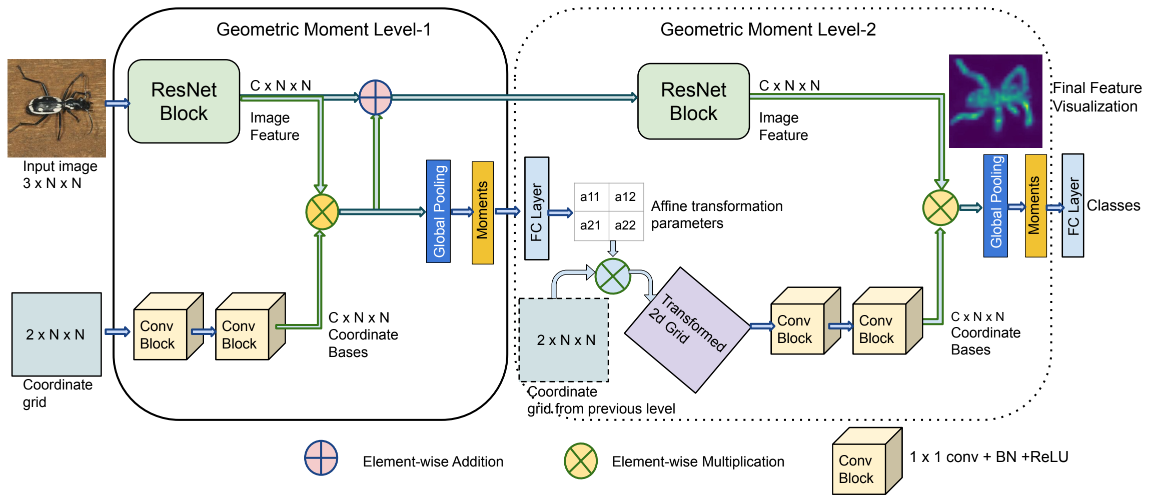


Figure 5.1: An Overview of Proposed Deep Geometric Moment (DGM) Framework for Image Classification Task. The Model Consists of Two Blocks: *Level-1* and *Level-2* That Consists of Two Pipelines : 1) CNN Based Image Feature Extraction and 2) Coordinate Bases Computation. the *Level-2* Block Can Be Repeated Number of times for Computing Moments, Similar to Depth Concept in Deep Networks.

CNNs are extremely good at capturing local context and texture information to discriminate images, even in a complex classification task, without an explicit 'shape' related operation. On the contrary, geometric moments can capture the shape information exceptionally well and provide discriminative cues in classifying images; however, their discrimination power is quite limited and generally requires a salient object over a homogeneous background. We advance a new type of architecture that blends the strengths of both approaches. We propose a deep geometric moment (DGM) model that uses geometric moments along with CNNs for classification by providing both shape and texture access. The geometric moment for a discrete 2D function is given by the discrete version of Eq. 5.1 :

$$m_{pq} = \sum_x \sum_y x^p y^q I(x, y) \tag{5.2}$$

In the traditional usage of moments in vision, the number and order of moments is an experimental design choice. Choosing the right number and order of moments depends on the underlying tasks; large moments are useful for image reconstruction, whereas, for image classification, higher-order moments are affected by noise and hence not very useful. Thus, selecting the correct moment orders is essential. In our method, we specify the required number of moments (in terms of feature dimension), but the exact basis functions and orders are learned by the networks end-to-end for specific tasks. Hence we will drop the subscript notation pq and use the superscript notation c indicating the feature channel number for moment m.

In our model defined by Eq. 5.3, we use CNNs to extract relevant object features from the given images and project them onto the learned coordinate bases *per channel*:

$$m^c = \frac{1}{N \times N} \sum_x \sum_y g^c(x, y) f^c(x, y), \tag{5.3}$$

where, $N \times N$ is the dimension of the image, $g^c(x, y)$ is a learnable 2D polynomial function, and $f^c(x, y)$ is image feature at coordinate location (x, y), and c refers to the channel dimension of the feature, given by the CNN from the image-feature stream (see top-stream in Fig. 5.1). Next, to account for varying locations, sizes, poses, and deformation, we allow our network to learn affine parameters to appropriately deform the 2D coordinate grid during moment computation.

In summary, the proposed model consists of three components: 1) *Coordinate base computation*: uses a 2D coordinate grid as input and generates the bases, 2) *Image feature computation*: obtains image features through ResNet blocks, and 3) *Affine transform estimation*: to transform the 2D coordinate grid to enable invariance learning. An overview of the DGM model is shown in Fig. 5.1. The architecture consists of *Level-1* and *Level-2* blocks, where *Level-1* is fixed, whereas *Level-2* can be replicated multiple times to create deeper networks.

Coordinate base computation: For computing bases, expressed as $g(x, y)$ in Eq. 5.3, a 2D coordinate grid is used as an input. The 2D coordinate grid is represented by $2 \times N \times N$, where $N \times N$ is the dimension of the input image. Each entry in the coordinate grid indicates the normalized 2D pixel locations. $g(x, y)$ in Eq. 5.3 is defined by a neural network that consists of two layers of 1×1 convolution layer followed by a batch-normalization and ReLU layer. This definition of $g(x, y)$ processes each location of the coordinate grid independently. We use only two convolution layers in our experiments, but one can use more layers to learn more complex or higher-order moments. The output bases are of dimension $C \times N \times N$ where C is the number of moments/channels.

Image feature computation: Referring back to the term $f(x, y)$ in Eq. 5.3, also shown as a ResNet block in Fig. 5.1, takes the image of dimension $3 \times N \times N$ as input, and outputs a feature of dimension $C \times N \times N$. $f(x, y)$ is implemented as a conventional ResNet block [49] with 3×3 filter kernel. Note that the geometric moments are insufficient for capturing local features in the image. In contrast, the CNNs with a kernel size of 3×3 or greater are very efficient in capturing local properties. Therefore, we use a kernel of size 3×3 in ResNet blocks, which is a common choice in state-of-the-art ResNet-based models. These image features from the ResNet block are then projected on the bases by performing element-wise multiplication, and moments are obtained by using global pooling on the projected features. Note that unlike the conventional definition of geometric moment Eq. 5.1 where the same 2D image is projected to each basis, in our method, different feature maps are projected to each basis. The projected feature map on the bases highlights the important region in the feature map.

Affine transformation estimation: We first use the canonical coordinate grid to compute the moments and predict the affine parameters using these moments. The prediction network consists of two fully-connected layers with a non-linear activation. The prediction network takes the canonical moments $(1 \times C)$ as input and outputs each feature channel's affine parameters $(C \times 6)$. Then, the 2D coordinate grid C_G is transformed according to: $C_G^{'} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \times C_G + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$, where, the 2×2 matrix is the predicted affine parameters, t_x and t_y are the predicted translation parameters and $C_G^{'}$ is the transformed 2D coordinate grid. We then use these transformed coordinate grids to generate new bases and compute new moments for each channel. Arguably, affine parameters are also limited in providing needed invariance and robustness, but this choice is efficient and leads to good performance.

5.3.1 DGM Classification Model

The proposed DGM network for the image classification task is shown in Fig. 5.1. It uses the computed coordinate bases, convolution network-based features, and affine transformation estimation and is trained end-to-end. The functionality of the model comprises of: 1) an image feature pipeline that transforms an image to features through ResNet blocks, and 2) a geometric moment pipeline that generates the bases and computes the affine parameters and moments. The proposed model does not use any pooling layer or reduce the spatial dimensions across the networks. This preserves the shape of the object, as opposed to pooling or reducing the spatial dimension that distorts the final reconstructed shape, limiting its interpretability. For simplicity, we also use the same number of feature channels in each ResNet layer.

As shown in Fig. 5.1, *Level-1* uses the canonical coordinate grid to generate bases and the ResNet block to generate features from the image. We then project this feature on the bases to compute the moments. The projected feature acts as an attention map and is added to the original feature. This feature map and geometric moments are then passed to *Level-2*.

The *Level-2* contains a ResNet block to process the features further. This level also predicts the affine parameters based on the moments from the previous level and transforms the coordinate grid to regenerate the bases. Fig. 5.1 shows only two levels, but one can repeat the *Level-2* block multiple times for added depth. The moments from the final level are used as input to the fully-connected layer to generate class probabilities for the classification task.

Feature Visualization: To visualize the shape awareness brought by the DGM approach, we describe a particular to visualize the learned features that highlight the object's shape. By the uniqueness theorem [57], moments can be used to reconstruct the original input, provided the bases are complete. In our case, our learned bases are under-complete. Using the moments as combination weights on the projected features given by:

$$V = \sum_c m^c (G^c \otimes F^c), \tag{5.4}$$

where, m^c is the moment, G^c is the basis, F^c is the image feature for channel c, and $\otimes$ is element-wise multiplication, we get a visualization of shape-related information in the features.

5.4 Experimental Results

We evaluate the proposed method for image classification on standard datasets: CIFAR-10, CIFAR-100 [75], and ILSVRC-2012 ImageNet [118] to validate the effectiveness of our model. The performance of our model is compared to a baseline model (ResNet model without pooling layers) and standard ResNet models [49] across classification accuracy (in %) and the number of parameters (in Million M). Along with the classification performance, we also compare the feature reconstruction qualitatively and semantic segmentation. We train all models under the same training hyperparameters. For CIFAR datasets, we train models up to 150 epochs with a batch size of 128; for ImageNet, we train for 100 epochs with a batch size of 256. We use SGD optimizer with $momentum = 0.9$, $weight\_decay = 5e^{-4}$, and cosine learning rate decay with an initial learning rate of 0.1.

5.4.1 How Many Levels Do We Need?

In this experiment, the ResNet block in the image feature pipeline consists of only 1×1 filter kernels, except the first Conv layer. There is no interaction between neighboring pixels in this setting; the only interaction is through geometric moments and affine transformation of the coordinate grid. Each level in our DGM model consists of a ResNet block with two ResNet layers, a coordinate bases generator defined with two convolutional layers, and two fully connected layers for affine parameters prediction.

Table 5.1: Performance Comparison of DGM Model with Increasing Levels on CIFAR Datasets.

Model	Params (M)	CIFAR 10 (%)	CIFAR 100 (%)
DGM Level-1	0.44	84.79	59.07
DGM Level-2	1.37	88.77	68.07
DGM Level-3	2.30	90.09	69.72
DGM Level-4	3.23	90.28	70.56
DGM Level-4 (w/o affine)	2.03	88.47	66.6

This setting helps us understand the overall contributions of the coordinate bases and the affine transformation.

Table 5.1 reports the DGM model's classification performance and the number of parameters as we increase the number of levels in the model. In *Level-1*, we use the canonical coordinate grid to generate the bases. The results show that DGM *Level-2* performs significantly better than DGM *Level-1* on both the CIFAR-10 and CIFAR-100 datasets. This performance improvement reflects the effectiveness of transforming the coordinate grid to regenerate better bases. We also see that the performance difference between the *Level-3* and *Level-4* model is minimal. In DGM *Level-4* w/o affine transform (last row), we do not transform the coordinate grid; hence the bases in this case, remain the same for every image. Without affine transformation, the model's performance drops, indicating the effectiveness of using affine transformation. Also, increasing the levels beyond 4 does not significantly improve the classification performance but is accompanied by a large increase in the number of parameters and computation; hence, we use only *Level-4* in all the experiments.

5.4.2 Comparison with Baseline ResNet Model

Table 5.2: Performance Comparison of Proposed DGM Model and Baseline ResNet Model (Without Pooling Layers) on CIFAR Datasets.

Model	Params (M)	CIFAR 10 (%)	CIFAR 100 (%)
ResNet-18 (w/o pooling)	9.62	94.78	76.93
DGM ResNet-18	11.61	**95.51**	**80.60**
ResNet-34 (w/o pooling)	18.94	95.46	78.42
DGM ResNet-34	21.06	**96.27**	**82.13**

Table 5.3: Performance Comparison of Proposed DGM Model with Baseline ResNet Model (Without Pooling Layers) on ImageNet Dataset.

Model	Params (M)	Accuracy (%)
ResNet-18 (w/o pooling)	9.89	68.42
DGM ResNet-18	11.88	**72.36**
ResNet-34 (w/o pooling)	19.20	73.34
DGM ResNet-34	21.32	**75.63**

The baseline model in this experiment is constructed in the same manner as our DGM model but without projection onto the coordinate bases. The baseline models are similar to standard ResNet models without pooling layers or reducing the spatial dimension. Without the reduction in the feature's spatial dimension, the receptive field of the filter kernels reduces and hence weakens the long-range dependency captured by the model. This experiment helps us establish that the moments effectively capture long-range dependency without reducing the spatial dimension of the features. We use global pooling on the features from the last ResNet layer of the baseline model to get the final feature vector for classification.

In Table 5.2, the baseline ResNet-18 and ResNet-34 models are based on the standard ResNet models with a constant number of feature channels (256) in every layer and without any pooling layers or reducing the spatial dimensions. DGM ResNet-18 and DGM ResNet-34 are also of 4 levels, with coordinate grid size of 32×32. Table 5.2 shows that the proposed DGM models perform much better than the baselines on both CIFAR datasets. The performance improvement validates the effectiveness of using coordinate bases pipeline.

For comparison on the ImageNet dataset, we use ResNet-18 and ResNet-34 type architectures on a grid size of 32×32. We divide the 256×256 image into 8×8 patches and use a linear embedding layer similar to Vision Transformer (ViT) [27] to reduce the spatial dimension to 32×32, followed by the DGM model. For the embedding layer, we use a convolution layer with 8×8 kernel and stride of 8. Both baseline and DGM models consist of 3×3 filter kernels and 256 feature channels and are trained under the same hyper-parameters. Table 5.3 shows that the proposed DGM model provides an improvement of $\sim 4\%$ (in the case of ResNet-18) over the baseline model with the same feature extraction pipeline. The performance of baseline models is less than the standard ResNet models. This performance reduction in the baseline models is mainly due to no pooling layers or reduction in features' spatial dimension, which results in a low receptive field. Our DGM model's performance is comparable to the standard ResNet model but without a spatial dimension reduction, showing that moments effectively capture the required long-range dependencies.

Table 5.4: Performance Improvements of DGM Models over Standard ResNet Model (with Pooling Layers) on CIFAR Datasets.

Model	Params (M)	CIFAR-10(%)	CIFAR-100(%)
ResNet-18	11.17	95.37	77.35
DGM ResNet-18	11.61	**95.51**	**80.60**
ResNet-34	21.33	95.58	78.83
DGM ResNet-34	21.06	**96.27**	**82.13**
MobileNet[56]	3.93	93.62	73.53
DGM MobileNet	4.50	**96.33**	**82.19**

Table 5.5: Performance Comparison of DGM Model with Standard ResNet Model (with Pooling Layers) on ImageNet Dataset.

Model	Params (M)	Accuracy (%)
ResNet-18	11.69	71.23
DGM ResNet-18	11.88	**72.36**
ResNet-34	21.80	74.58
DGM ResNet-34	21.32	**75.63**
ResNet-50	25.56	76.92
DGM ResNet-50	23.51	**77.06**
MobileNet[56]	4.20	70.66
DGM MobileNet	4.76	**72.69**

5.4.3 Comparison with Standard ResNet Model

Table 5.4 and 5.5 compare the proposed DGM model with conventional ResNet models [49]. The number of feature channels in the standard ResNet model is $(64, 128, 256, 512)$; whereas, in our DGM model, the number of feature channels is 256 in ResNet-18 and Resnet-34 and 512 in ResNet-50 across all levels. The naming of our DGM model is based on the number of layers used in the image feature pipeline (similar to the standard ResNet model naming convention). The standard ResNet models use pooling layers to reduce the spatial dimensions up to 8×8 in the final layer, whereas, in our DGM model, the final feature's spatial dimension is 32×32.

Table 5.4 shows that the proposed DGM models perform better than the standard ResNet models on both CIFAR-10 and CIFAR-100 datasets. The DGM model improves the accuracy of $\sim 1\%$ on CIFAR-10 and $\sim 3\%$ on CIFAR-100. Table 5.5 compares the DGM model with the standard ResNet model on the ImageNet dataset. We observe that our DGM model is better than the standard ResNet model while using a similar number of parameters but without using any pooling layers.

The computation cost in our DGM model is much higher than conventional ResNet, which is attributable to the fact that we do not reduce the spatial dimension of the image features. However, one can reduce the computation cost in the image pipeline using the channel-wise convolution [56, 124]. In Table 5.4 and 5.5, DGM MobileNet has very less number of parameters, but performs comparable to DGM model with conventional ResNet layers.

5.4.4 Feature Visualization

A major advantage of keeping a higher spatial dimension of features is interpretability at different levels. The feature vectors can be easily visualized by a reconstruction step as given by Eq. 5.4. Fig. 5.2 compares *Level-4* visualization as a heatmap for a few randomly selected images from the ImageNet dataset for our DGM model against baseline ResNet and standard ResNet models. The visualization for the baseline model is just a weighted sum of the final Conv layer activation based on the global feature. For the standard ResNet model, we use GradCAM [128]. We also compare

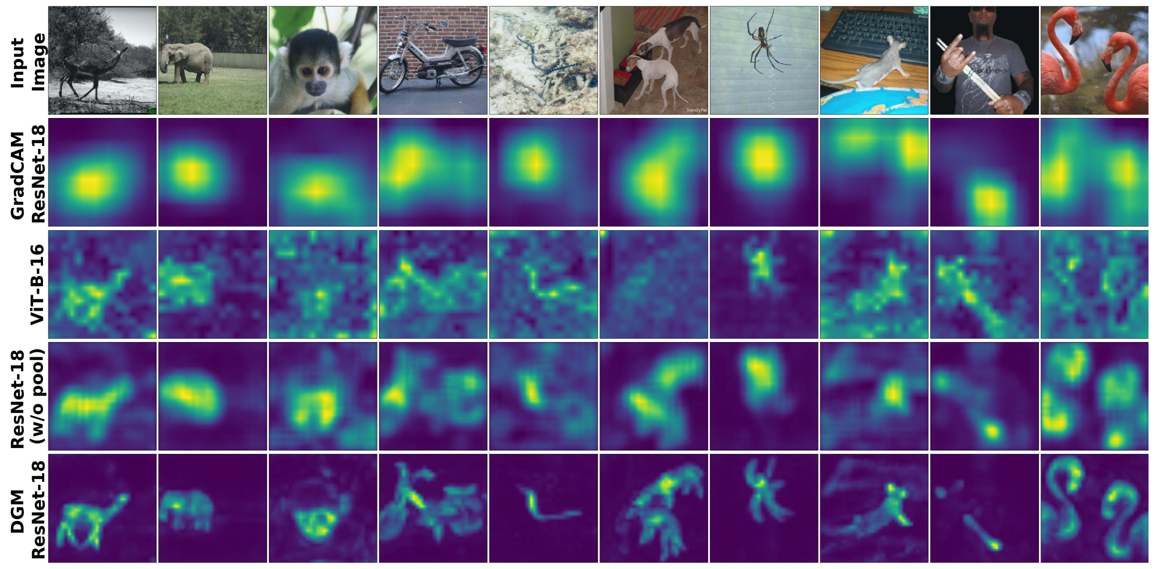


Figure 5.2: Feature Visualization of Different Models on ImageNet. For the Standard ResNet Model, We Use Gradcam for Visualization. We Also Compare Our Visualization with the Vision Transformer [27] (Vit-B-16) Attention Map. Note That Our DGM Model Produces a Very Sharp Object Shape.

our visualization with the current attention-based Vision Transformer [27] (ViT-B-16) model, which is pre-trained on the ImageNet-21K and finetuned on ImageNet-1K datasets. As shown in Fig. 5.2, the GradCAM visualizations of the standard ResNet-18 model generate a blob-like shape around the critical region in the image, with no discernible object shape. While it gets better for the baseline ResNet-18 model, the heatmaps are still diffuse, and the shapes are not very distinct. However, with DGM model, object shapes are crisp, with improved classification accuracies. Also, our heat map is much sharper than the vision transformer attention map (Vit-B-16).

Additionally, in the DGM model, we can visualize features at different levels providing much better-debugging capability, as shown in Fig. 5.3. At initial levels, the heatmap is noisy, and the model is not able to able to separate the object from the background clearly as compared to higher levels.

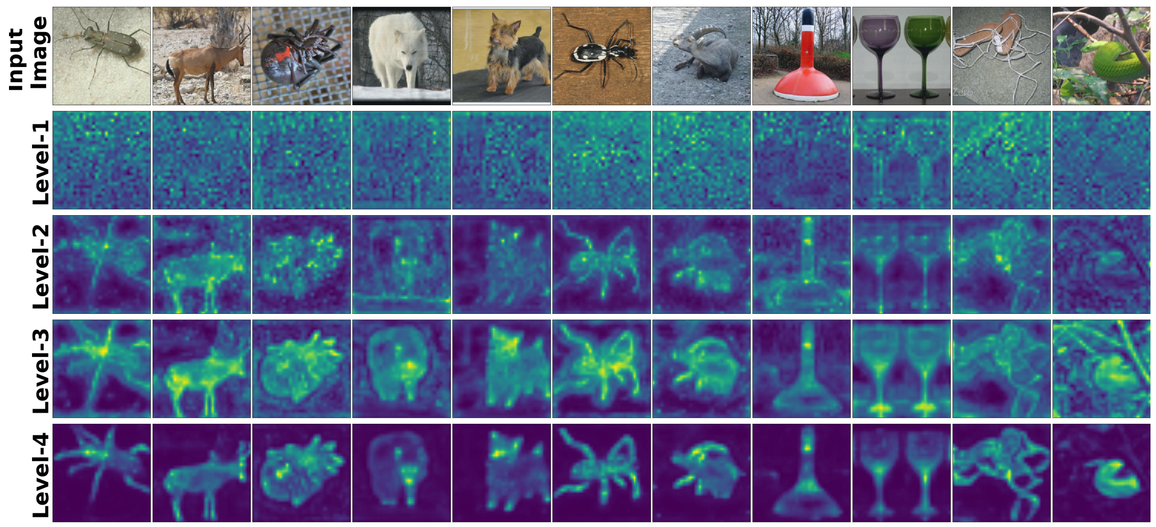


Figure 5.3: Visualization at Different Levels for DGM ResNet-34 Model on the ImageNet Dataset. We Note That at Higher Levels Our Model Is Able to Separate the Background Information from the Object's Shape Compared to Initial Levels.

5.4.5 Finetuning

For DGM finetuning, we only need to retrain the coordinate bases pipeline, and the final classifier layer, while freezing the image feature exaction pipeline. The coordinate bases pipeline contains significantly fewer parameters and requires less computation. We finetune our ImageNet pre-trained DGM model for only 30 epochs on CIFAR-10 and CIFAR-100 datasets. The network is finetuned using a SGD optimizer with a cosine decay learning rate and an initial learning rate of 0.01. Table 5.6 shows the accuracy of the finetuned model on the CIFAR-10 and CIFAR-100 datasets. The performance of the finetuned model drops as compared to DGM trained from scratch, but it performs equally well to the standard ResNet model trained from scratch.

5.4.6 Performance under Color Distortion

We evaluate the effect of color distortions on DGM performance by testing it on ImageNet-C [50]. The ImageNet pretrained DGM model used for this experiment does not use any color augmentation during training, making it a sufficiently challenging

Table 5.6: Classification Performance on Fine-tuning of Pre-trained ImageNet DGM Model on CIFAR Datasets.

Model	Params (M)	CIFAR 10(%)	CIFAR 100(%)
DGM ResNet-18	11.69	93.79	75.83
DGM ResNet-34	21.32	94.01	77.51
DGM MobileNet	4.76	93.87	75.92

Table 5.7: Mean Corruption Error (mCE) Comparison of DGM with Standard ResNet Model on ImageNet-C Dataset.

Model	Params (M)	Clean ↑ Acc. (%)	mCE ↓ (%)
ResNet-50	25.56	76.92	74.97
DGM ResNet-50	23.51	77.06	**71.74**

task. The DGM ResNet-50 and GradCAM ResNet-50 visualization for two distortions is shown in Fig. 5.4. The figure shows that our model captures the object shape very well under different challenging distortions like fog and blur. The GradCAM heatmap for the ResNet-50 is not very consistent across distortions as compared to our DGM model. For quantitative performance, we use the mean Corruption Error (mCE) metric (lower is better) [50]. We choose our DGM model such that it performs comparably to the standard ResNet-50 model on clean images but with fewer parameters. Table 5.7 shows DGM model provides an improvement of 3.2 points on the corrupted images.

5.4.7 Semantic Image Segmentation

The DGM model is evaluated on the PASCAL VOC 2012 [29] and Cityscapes [20] semantic segmentation benchmark datasets. We use a *Level-5* DGM model with

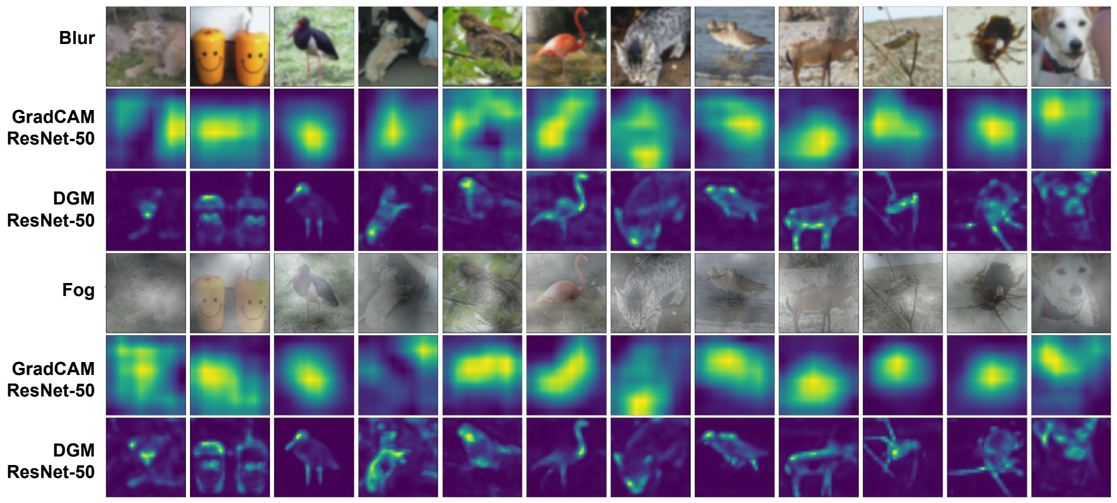


Figure 5.4: Visualization from DGM and Standard ResNet Model under Two Different Color Distortion (Blur (1^{st} Row) and Fog (4^{th} Row) from ImageNet-C). Our Model (3^{rd} and 6^{th} Row), Is Able to Produce Consistent Shape Across Different Distortions Compared to Standard ResNet (2^{nd} and 5^{th} Row).

ResNet-50 as the image-feature pipeline pretrained on the ImageNet dataset. The number of parameters in the *Level-5* DGM ResNet-50 is almost identical to the standard ResNet-50 model ($\sim$ 25M). We use the same training hyperparameters as in the DeepLabv3+ model [15]. We test the effectiveness of our model with two different segmentation heads; first, with just two 1×1 Conv layers as segmentation head that takes the final 2D features rescaled by factor of 4, and second, DeepLabv3+ segmentation head, which consists of atrous convolution with different rates to capture long range dependencies. In Table 5.8, we observe that our model performs comparable to the standard DeepLabv3+ model, even with a very simple segmentation head on both datasets. This shows that geometric moments are effective in capturing long range dependencies. Our model shows improvements of 1.5% points on Pascal VOC and 0.7% points on Cityscapes Val sets compared to the standard ResNet model.

Table 5.8: Semantic Segmentation Perfomance on PASCAL VOC 2012 [29] and Cityscapes [20] Val Set in Terms of Mean Intersection over Union (mIoU).

Backbone	Segmentation head	PASCAL VOC (mIoU)	Cityscapes (mIoU)
ResNet-50	1×1 Conv	68.59	71.92
DGM-ResNet-50	1×1 Conv	**78.43**	**74.77**
ResNet-50	DeepLabv3+	78.36	75.34
DGM-ResNet-50	DeepLabv3+	**79.89**	**76.03**

5.5 Conclusion

In this work, we propose a geometric moment-based deep learning model that explicitly captures the shape-related information in an end-to-end learnable fashion. The DGM model improves the interpretability of features while also learning discriminative features. The quantitative and qualitative results on standard image classification and segmentation datasets show that our method performs better than the corresponding baseline and standard ResNet models. In addition, the DGM model provides easy interpretability at different levels while also providing ease of finetuning on a given dataset. Further, our model captures the object's shape even under extreme affine and color aberrations while performing better than existing approaches. We believe that DGM can improve the performance of other vision tasks, such as object detection and generation, and can be generalized to other modalities like video, RGBD, and volumetric data.

Chapter 6

POLYNOMIAL IMPLICIT NEURAL REPRESENTATIONS FOR LARGE DIVERSE DATASETS

6.1 Introduction

Deep learning-based generative models are a very active area of research with numerous advancements in recent years [73, 42, 25]. Most widely, generative models are based on convolutional architectures. However, recent developments such as implicit neural representations (INR) [98, 133] represent an image as a continuous function of its coordinate locations, where each pixel is synthesized independently. Such a function is approximated by using a deep neural network. INR provides flexibility for easy image transformations and high-resolution up-sampling through the use of a coordinate grid. Thus, INRs have become very effective for 3D scene reconstruction and rendering from very few training images [98, 97, 9, 96, 163]. However, they are usually trained to represent a single given scene, signal, or image. Recently, INRs have been implemented as a generative model to generate entire image datasets [6, 136]. They perform comparably to CNN-based generative models on perfectly curated datasets like human faces [68]; however, they have yet to be scaled to large, diverse datasets like ImageNet [22].

INR generally consists of a positional encoding module and a multi-layer perceptron model (MLP). The positional encoding in INR is based on sinusoidal functions, often referred to as Fourier features. Several methods [98, 133, 142] have shown that using MLP without sinusoidal positional encoding generates blurry outputs, i.e., only preserves low-frequency information. Although, one can remove the positional encoding by replacing the ReLU activation with a periodic or non-periodic activation function in the MLP [133, 111, 18]. However, in INR-based GAN [6], using a periodic activation function in MLP leads to subpar performance compared to positional encoding with ReLU-based MLP.

Sitzmann et al. [133] demonstrate that ReLU-based MLP fails to capture the information contained in higher derivatives. This failure to incorporate higher derivative information is due to ReLU's piece-wise linear nature, and second or higher derivatives of ReLU are typically zero. This can be further interpreted in terms of the Taylor series expansion of a given function. The higher derivative information of a function is included in the coefficients of a higher-order polynomial derived from the Taylor series. Hence, the inability to generate high-frequency information is due to the ineffectiveness of the ReLU-based MLP model in approximating higher-order polynomials.

Sinusoidal positional encoding with MLP has been widely used, but the capacity of such INR can be limiting for two reasons. First, the size of the embedding space is limited; hence only a finite and fixed combination of periodic functions can be used, limiting its application to smaller datasets. Second, such an INR design needs to be mathematically coherent. These INR models can be interpreted as a non-linear combination of periodic functions where periodic functions define the initial part of the network, and the later part is often a ReLU-based non-linear function. Contrary to this, classical transforms (Fourier, sine, or cosine) represent an image by a linear summation of periodic functions. However, using just a linear combination of the positional embedding in a neural network is also limiting, making it difficult to represent large and diverse datasets. Therefore, instead of using periodic functions, this work models an image as a polynomial function of its coordinate location.

The main advantage of polynomial representation is the easy parameterization of polynomial coefficients with MLP to represent large datasets like ImageNet. However, conventionally MLP can only approximate lower-order polynomials. One can use a polynomial positional embedding of the form $x^p y^q$ in the first layer to enable the MLP to approximate higher order. However, such a design is limiting, as a fixed embedding size incorporates only fixed polynomial degrees. In addition, we do not know the importance of each polynomial degree beforehand for a given image.

Hence, we do not use any positional encoding, but we progressively increase the degree of the polynomial with the depth of MLP. We achieve this by element-wise multiplication between the feature and affine transformed coordinate location, obtained after every ReLU layer. The affine parameters are parameterized by the latent code sampled from a known distribution. This way, our network learns the required polynomial order and represents complex datasets with considerably fewer trainable parameters. In particular, the key highlights are summarized as follows:

- We propose a Poly-INR model based on polynomial functions and design a MLP model to approximate higher-order polynomials.
- Poly-INR as a generative model performs comparably to the state-of-the-art CNN-based GAN model (StyleGAN-XL [127]) on the ImageNet dataset with $3-4\times$ fewer trainable parameters (depending on output resolution).
- Poly-INR outperforms the previously proposed INR models on the FFHQ dataset [68], using a significantly smaller model.
- We present various qualitative results demonstrating the benefit of our model for interpolation, inversion, style-mixing, high-resolution sampling, and extrapolation.

6.2 Related Work

Implicit neural representations: INRs have been widely adopted for 3D scene representation and synthesis [135, 97, 98]. Following the success of NeRF [98], there has been a large volume of work on 3D scene representation from 2D images [160, 163, 134, 96, 109, 62, 9]. They have also been used for semantic segmentation [36], video [105, 157, 37], audio [37], and time-series modeling [34]. INRs have also been used as a prior for inverse problems [133, 112]. However, most INR approaches either use a sinusoidal positional encoding [98, 142] or a sinusoidal activation function [133], which limits the model capacity for large dataset representation. In our work, we represent our Poly-INR model as a polynomial function without using any positional encoding.

GANs: have been widely used for image generation and synthesis tasks [42]. In recent work, several improvements have been proposed[110, 68, 100, 8, 44] over the original architecture. For example, the popularly used StyleGAN [68] model uses a mapping network to generate style codes which are then used to modulate the weights of the Conv layers. StyleGAN improves image fidelity, as well as enhances inversion [146] and image editing capabilities [47]. StyleGAN has been scaled to large datasets like ImageNet [127], using a discriminator which uses projected features from a pre-trained classifier [125]. More recently, transformer-based models have also been used as generators [172, 81]; however, the self-attention mechanism is computationally costly for achieving higher resolution. Unlike these methods, our generator is free of convolution, normalization, and self-attention mechanisms and only uses ReLU and Linear layers to achieve competitive results, but with far fewer parameters.

GANs + coordinates: INRs have also been implemented within generative models. For example, CIPS [6] uses Fourier features and learnable vectors for each spatial location as positional encoding and uses StyleGAN-like weight modulation for the linear layers in the MLP. Similarly, INR-GAN [136] proposes a multi-scale generator model where a hyper-network determines the parameters of the MLP. INR-GAN has been further extended to generate an ‘infinite’-size continuous image using anchors [137]. However, these INR-based generative models have only shown promising results on smaller datasets. Our work scales easily to large datasets like ImageNet owing to the significantly fewer parameters in Poly-INR.

Other approaches have combined convolution-based generation with coordinate-based features. For example, the Local Implicit Image Function (LIIF) [17] and Spherical Local Implicit Image Function (SLIIF) [162] use a CNN-based backbone to generate feature vectors corresponding to each coordinate location. Arbitrary-scale image synthesis [104] uses a multi-scale convolution-based generator model with scale-aware position embedding to generate scale-consistent images. StyleGAN model, further extended by [67] (StyleGAN-3) to use coordinate location-based Fourier features. In addition, StyleGAN-3 uses filter kernels equivariant to the coordinate grid's translation and rotation. However, the rotation equivariant version of the StyleGAN-3 model fails to scale to ImageNet dataset, as reported in [127]. Instead of using convolution layers, the Poly-INR only uses linear and ReLU layers.

Relation to classical geometric moment: Polynomial functions have been explored earlier in the form of geometric moments for image reconstruction [57, 143, 55, 33]. Unlike the Fourier transform, which uses the sinusoidal functions as the basis, the geometric moment method projects the 2D image on a polynomial basis of the form $x^p y^q$ to compute the moment of order $p + q$. The moment matching method [143] is generally used for image reconstruction from given finite moments. In moment matching, the image is assumed to be a polynomial function, and the coefficients of the polynomial are defined to match the given finite moments. Similar to geometric moments, we also represent images on a polynomial basis; however, our polynomial coefficients are learned end-to-end and defined by a deep neural network.

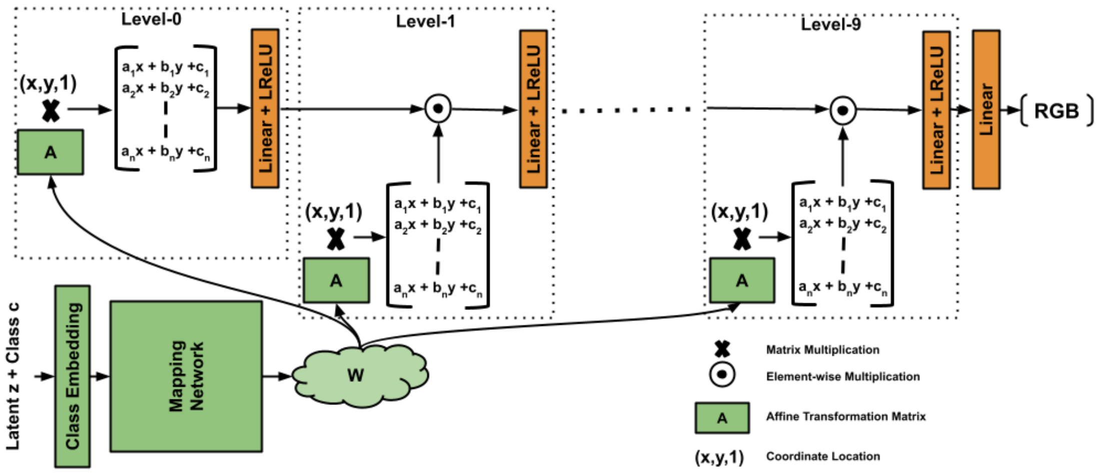


Figure 6.1: Overview of Our Proposed Polynomial Implicit Neural Representation (Poly-INR) Based Generator Architecture. Our Model Consists of Two Networks: 1) Mapping Network, Which Generates the Affine Parameters from the Latent Code z, and 2) Synthesis Network, Which Synthesizes the RGB Value for the given Pixel Location. Our Poly-INR Model Is Defined Using Only Linear and ReLU Layers End-to-end.

6.3 Method

We are interested in a class of functions which represent an image in the form:

$$G(x, y) = g_{00} + g_{10}x + g_{01}y + \ldots + g_{pq}x^p y^q, \tag{6.1}$$

where, (x, y) is the normalized pixel location sampled from a coordinate grid of size $(H \times W)$, while the coefficients of the polynomial (g_{pq}) are parameterized by a latent vector z sampled from a known distribution and are independent of the pixel location. Therefore, to form an image, we evaluate the generator G for all pixel locations (x, y) for a given fixed z:

$$I = \{G(x, y; z) \,|\, (x, y) \in CoordinateGrid(H, W)\}, \tag{6.2}$$

where, $CoordinateGrid(H, W) = \{(\frac{x}{W-1}, \frac{y}{H-1}) \,|\, 0 \le x < W, 0 \le y < H\}$. By sampling different latent vectors z, we generate different polynomials and represent images over a distribution of real images.

Our goal is to learn the polynomial defined by Eq. 6.1 using only Linear and ReLU layers. However, the conventional definition of MLP usually takes as input the coordinate location, processed by a few Linear and ReLU layers. This definition of INR can only approximate low-order polynomials and hence only generates low-frequency information. Although, one can use a positional embedding consisting of polynomials of the form $x^p y^q$ to approximate a higher-order polynomial. However, this definition of INR is limiting since a fixed-size embedding space can contain only a small combination of polynomial orders. Furthermore, we do not know which polynomial order is essential to generate the image beforehand. Hence, we progressively increase the polynomial order in the network and let it learn the required orders. We implement this by using element-wise multiplication with the affine-transformed coordinate location at different levels, shown in Fig 6.1. Our model consists of two parts: 1) **Mapping network**, which takes the latent code z and maps it to affine parameters space $\mathbf{W}$, and 2) **Synthesis network**, which takes the pixel location and generates the corresponding RGB value.

Mapping Network: The mapping network takes the latent code $z \in \mathbb{R}^{64}$ and maps it to the space $\mathbf{W} \in \mathbb{R}^{512}$. Our model adopts the mapping network used in [127]. It consists of a pre-trained class embedding, which embeds the one hot class label into a 512 dimension vector and concatenates it with the latent code z. Then the mapping network consists of an MLP with two layers, which maps it to the space $\mathbf{W}$. We use this $\mathbf{W}$ to generate affine parameters by using additional linear layers; hence we call our $\mathbf{W}$ as affine parameters space.

Synthesis network: The synthesis network generates the RGB ($\mathbb{R}^3$) value for the given pixel location (x, y). As shown in Fig. 6.1, the synthesis network consists of multiple levels; at each level, it receives the affine transformation parameters from the mapping network and the pixel coordinate location. At *level*-0, we affine transform the coordinate grid and feed it to a Linear layer followed by a Leaky-ReLU layer with $negative\_slope = 0.2$. At later levels, we do element-wise multiplication between the feature from the previous level and the affine-transformed coordinate grid, and then feed it to Linear and Leaky-ReLU layers. With the element-wise multiplication at each level, the network has the flexibility to increase the order for x or y coordinate position, or not to increase the order by keeping the affine transformation coefficient $a_j = b_j = 0$. In our model, we use 10 levels, which is sufficient to generate large datasets like ImageNet. Mathematically, the synthesis network can be expressed as follows:

$$G_{syn} = \ldots \sigma(W_2((A_2X) \odot \sigma(W_1((A_1X) \odot \sigma(W_0(A_0X)))))), \tag{6.3}$$

where $X \in \mathbb{R}^{3\times HW}$ is the coordinate grid of size $H \times W$ with an additional dimension for the bias, $A_i \in \mathbb{R}^{n\times 3}$ is the affine transformation matrix from the mapping network for *level-i*, $W_i \in \mathbb{R}^{n\times n}$ is the weight of the linear layer at *level-i*, σ is the Leaky-ReLU layer and $\odot$ is element-wise multiplication. Here n is the dimension of the feature channel in the synthesis network, which is the same for all levels. For large datasets like ImageNet, we choose the channel dimension $n = 1024$, and for smaller datasets like FFHQ, we choose $n = 512$. Note that with this definition, our model only uses Linear and ReLU layers end-to-end and synthesizes each pixel independently.

Relation to StyleGAN: StyleGANs [68, 69, 67] can be seen as a special case of our formulation. By keeping the coefficients (a_j,b_j) in the affine transformation matrix of x and y coordinate location equal to zero, the bias term c_j would act as a style code. However, our affine transformation adds location bias to the style code, rather than just using the same style code for all locations in StyleGAN models. This location bias makes the model very flexible in applying a style code only to a specific image region, making it more expressive. In addition, our model differs from the StyleGANs in many aspects. First, our method does not use weight modulation/demodulation or normalizing [69] tricks. Second, our model does not employ low-pass filters or convolutional layers. Finally, we do not inject any spatial noise into our synthesis network. We can also use these tricks to improve the model's performance further. However, our model's definition is straightforward compared to other GAN models.

Table 6.1: Quantitative Comparison of Poly-INR Method with CNN-based Generative Models on ImageNet Datasets. (d) Compares the Number of Parameters Used in All Models at Various Resolutions. The Results for Existing Methods Are Quoted from the StyleGAN-XL Paper.

(a) ImageNet 128 × 128

Model	**FID** ↓	**sFID** ↓	**rFID** ↓	**IS** ↑	**Pr** ↑	**Rec** ↑
BigGAN	6.02	7.18	6.09	145.83	0.86	0.35
CDM	3.52	-	-	128.80	-	-
ADM	5.91	5.09	13.29	93.31	0.70	0.65
ADM-G	2.97	5.09	3.80	141.37	0.78	0.59
StyleGAN-XL	1.81	3.82	1.82	200.55	0.77	0.55
Poly-INR	2.08	3.93	2.76	179.64	0.70	0.45

(b) ImageNet 256 × 256

Model	**FID** ↓	**sFID** ↓	**rFID** ↓	**IS** ↑	**Pr** ↑	**Rec** ↑
BigGAN	6.95	7.36	75.24	202.65	0.87	0.28
ADM	10.94	6.02	125.78	100.98	0.69	0.63
ADM-G	3.94	6.14	11.86	215.84	0.83	0.53
DiT-XL/2-G	2.27	4.60	-	278.54	0.83	0.57
StyleGAN-XL	2.30	4.02	7.06	265.12	0.78	0.53
Poly-INR	2.86	4.37	7.79	241.43	0.71	0.39

(c) ImageNet 512 × 512

Model	**FID** ↓	**sFID** ↓	**rFID** ↓	**IS** ↑	**Pr** ↑	**Rec** ↑
BigGAN	8.43	8.13	312.00	177.90	0.88	0.29
ADM	23.24	10.19	561.32	58.06	0.73	0.60
ADM-G	3.85	5.86	210.83	221.72	0.84	0.53
DiT-XL/2-G	3.04	5.04	-	240.82	0.84	0.54
StyleGAN-XL	2.41	4.06	51.54	267.75	0.77	0.52
Poly-INR	3.81	5.06	54.31	267.44	0.70	0.34

(d) Number of parameters in millions (M)

Model	**64^2**	**128^2**	**256^2**	**512^2**
BigGAN	-	141.0	164.3	164.7
ADM	296.0	422.0	554.0	559.0
DiT-XL	-	-	675.0	675.0
StyleGAN-XL	134.4	158.7	166.3	168.4
Poly-INR	46.0	46.0	46.0	46.0

6.4 Experiments

The effectiveness of our model is evaluated on two datasets: 1) ImageNet [22] and 2) FFHQ [68]. The ImageNet dataset consists of $1.2M$ images over $1K$ classes, whereas the FFHQ dataset contains $\sim 70K$ images of curated human faces. All our models have 64 dimensional latent space sampled from a normal distribution with

Table 6.2: Quantitative Comparison of Poly-INR Method with CNN and INR-based Generative Models on FFHQ Dataset at 256×256.

Model	**params (M)**	**FID ↓**	**Inference Time (sec/img)**
StyleGAN2	30.0	3.83	0.016
StyleGAN-XL	67.9	2.19	0.047
CIPS	45.9	4.38	0.067
INR-GAN	72.4	4.95	0.024
Poly-INR	13.6	2.72	0.054

mean 0 and standard deviation 1. The affine parameters space $\mathbf{W}$ of the mapping network is 512 dimensions, and the synthesis network consists of 10 levels with feature dimension $n = 1024$ for the ImageNet and $n = 512$ for FFHQ. We follow the training scheme of the StyleGAN-XL method [127] and use a projected discriminator based on the pre-trained classifiers (DeiT [145] and EfficientNet [141]) with an additional classifier guidance loss [25].

We train our model progressively with increasing resolution, i.e., we start by training at low resolution and continue training with higher resolutions as training progresses. Since the computational cost is less at low resolution, the model is trained for large number of iterations, followed by training for high resolution. Since the model is already trained at low resolution, fewer iterations are needed for convergence at high resolution. However, unlike StyleGAN-XL, which freezes the previously trained layers and introduces new layers for higher resolution, Poly-INR uses a fixed number of layers and trains all the parameters at every resolution.

6.4.1 Quantitative Results

We compare our model against CNN-based GANs (BigGAN [12] and StyleGAN-XL [127]) and diffusion models (CDM [53], ADM, ADM-G [25], and DiT-XL [107]) on the ImageNet dataset. We also report results on the FFHQ dataset for INR-based GANs (CIPS [6] and INR-GAN [136]) as they do not train models on ImageNet.

Quantitative metrics: We use Inception Score (IS) [122], Frechet Inception Distance (FID) [51], Spatial Frechet Inception Distance (sFID) [102], random-FID (rFID) [127], precision (Pr), and recall (Rec) [78]. IS (higher the better) quantifies the quality and diversity of the generated samples based on the predicted label distribution by the Inception network but does not compare the distribution of the generated samples with the real distribution. The FID score (lower the better) overcomes this drawback by measuring the Frechet distance between the generated and real distribution in the Inception feature space. Further, sFID uses higher spatial features from the Inception network to account for the spatial structure of the generated image. Like StyleGAN-XL, we also use the rFID score to ensure that the network is not just optimizing for IS and FID scores. We use the same randomly initialized Inception network provided by [127]. In addition, we also compare our model on the precision and recall metric (higher the better) that measures how likely the generated sample is from the real distribution.

Table 6.1 summarizes the results on the ImageNet dataset at different resolutions. The results for existing methods are quoted from the StyleGAN-XL paper. We observe that the performance of the proposed model is third best after DiT-XL and StyleGAN-XL on the FID and IS metrics. The proposed model outperforms the ADM and BigGAN models at all resolutions and performs comparably to the StyleGAN-XL at 128×128 and 256×256. We also observe that with the increase in image size, the

FID score for Poly-INR drops much more than StyleGAN-XL. The FID score drops more because our model does not add any additional layers with the increase in image size. For example, the StyleGAN-XL uses 134.4M parameters at 64×64 and 168.4M at 512×512, whereas Poly-INR uses only 46.0M parameters at every resolution, as reported in Table 6.1(d). The table shows that our model performs comparably to the state-of-the-art CNN-based generative models, even with significantly fewer parameters. On precision metric, the Poly-INR method performs comparably to other methods; however, the recall value is slightly lower compared to StyleGAN-XL and diffusion models at higher resolution. Again, this is due to the small model size, limiting the model's capacity to represent much finer details at a higher resolution.

We also compare the proposed method with other INR-based GANs: CIPS and INR-GAN on the FFHQ dataset. Table 6.2 shows that the proposed model significantly outperforms these models, even with a small generator model. Interestingly the Poly-INR method outperforms the StyleGAN-2 and performs comparable to StyleGAN-XL, using significantly fewer parameters. Table 6.2 also reports the inference speed of these models on a Nvidia-RTX-6000 GPU. StyleGANs and INR-GAN use a multi-scale architecture, resulting in faster inference. In contrast, CIPS and Poly-INR models perform all computations at the same resolution as the output image, increasing the inference time.

6.4.2 Qualitative Results

Fig. 6.2 shows images sampled at different resolutions by the Poly-INR model trained on 512×512. We observe that our model generates diverse images with very high fidelity. Even though the model does not use convolution or self-attention layers, it generates realistic images over datasets like ImageNet. In addition, the model provides flexibility to generate images at different scales by changing the size of the

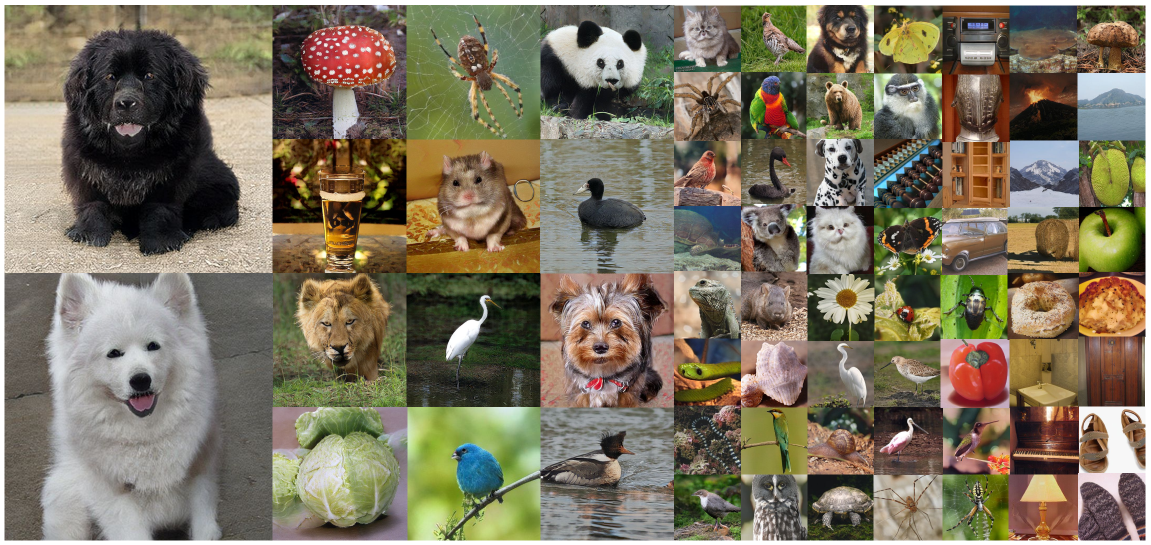


Figure 6.2: Samples Generated by Our Poly-INR Model on the ImageNet Dataset at Various Resolutions. Our Model Generates Images with High Fidelity Without Using Convolution, Upsample, or Self-Attention Layers, i.e., No Interaction Between the Pixels.

coordinate grid, making the model efficient if low-resolution images are needed for a downstream task. In contrast, CNN-based models generate images only at the training resolution due to the non-equivariant nature of the convolution kernels to image scale.

We also compare the quality of images generated by Poly-INR model against state-of-the-art CNN-based StyleGAN-XL model for different classes. Fig. 6.3 shows examples of images generated from different classes for the models trained on ImageNet at 256×256. The Poly-INR generates samples qualitatively similar to the StyleGAN-XL model but without using any convolution or self-attention layers. We also provide qualitative comparison of Poly-INR model against previously proposed INR-based generative models like CIPS and INR-GAN. Fig. 6.4 shows samples generated by the three models trained on the FFHQ dataset at 256×256. Our Poly-INR model generates qualitatively better samples than the CIPS and INR-GAN using significantly fewer parameters.

Heat-map visualization: Fig. 6.5 visualizes the heat-map at different levels of our

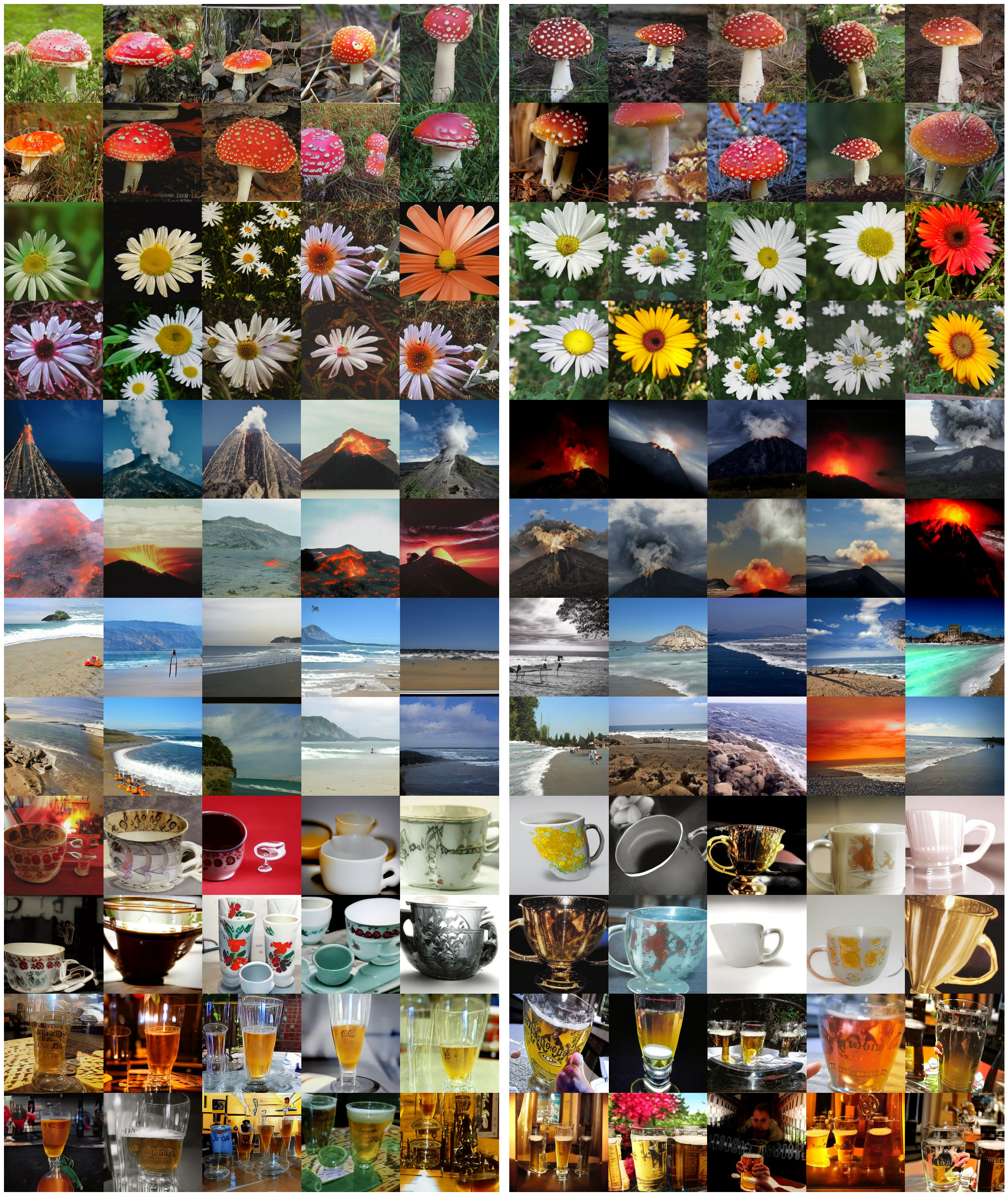


Figure 6.3: Qualitative Comparison Between StyleGAN-XL (Left Column) and Poly-INR (Right Column). Classes from Top to Bottom: Agaric, Daisy, Volcano, Seashore, Cup, and Beer Glass.

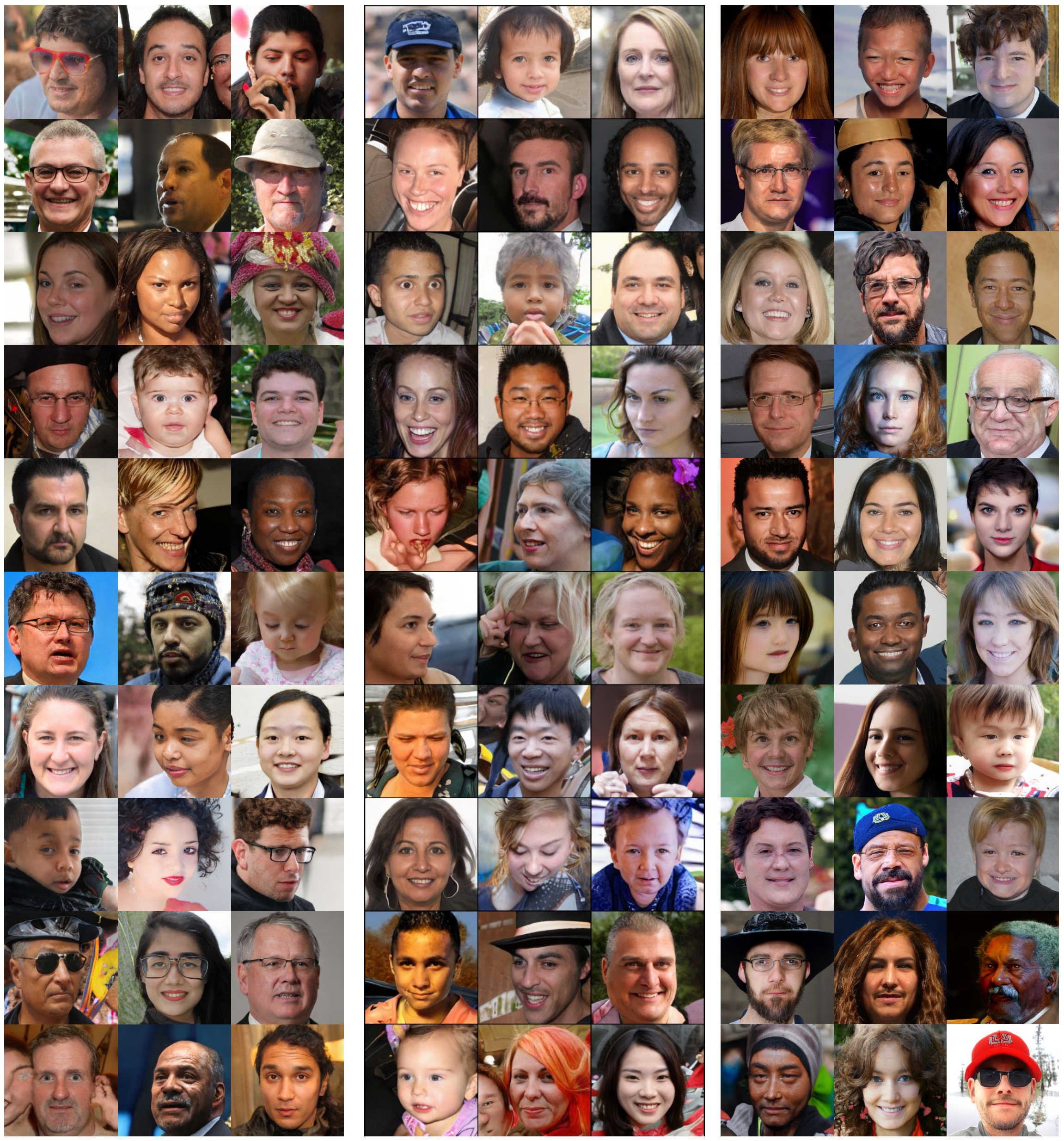


Figure 6.4: Qualitative Comparison Between INR-GAN (Left Column), CIPS (Middle Column), and Poly-INR (Right Column) on FFHQ Dataset at 256×256.

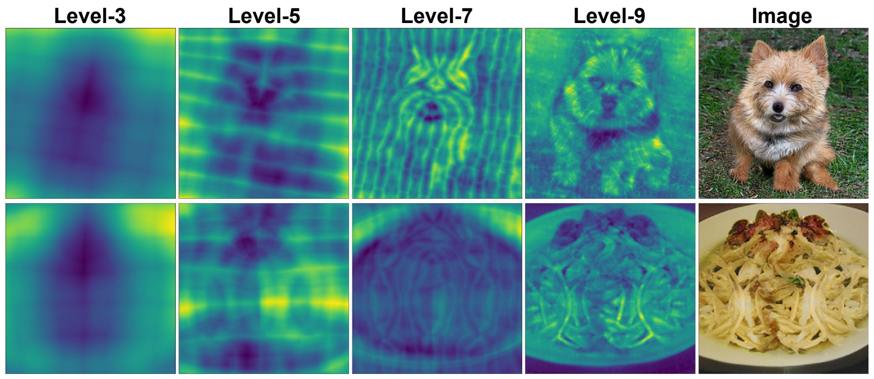


Figure 6.5: Heat-Map Visualization at Different Levels of the Synthesis Network. At Initial Levels, the Model Captures the Basic Shape of the Object, and at Higher Levels, the Image's Finer Details Are Captured.

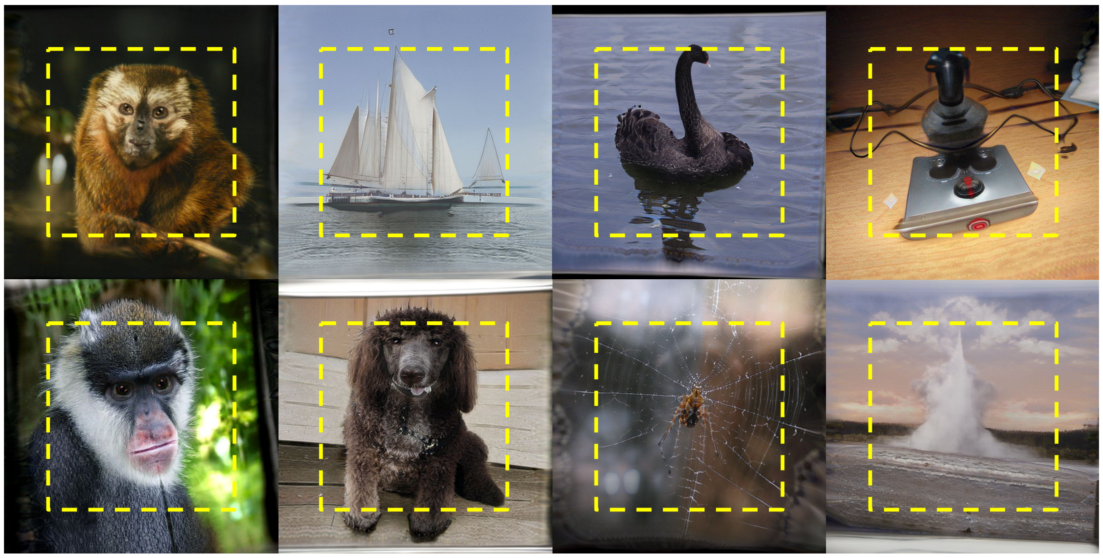


Figure 6.6: Few Example Images Showing Extrapolation Outside the Image Boundary (Yellow Square). The Poly-INR Model Is Trained to Generate Images on the Coordinate Grid $[0, 1]^2$. For Extrapolation, We Use the Grid Size $[-0.25, 1.25]^2$. Our Model Generates Continuous Image Outside the Conventional Boundary.

synthesis network. To visualize a feature as a heat-map, we first compute the mean along the spatial dimension of the feature and use it as a weight to sum the feature along the channel dimension. In the figure, we observe that in the initial levels (0-3), the model forms the basic structure of the object. Meanwhile, in the middle levels (4-6), it captures the object's overall shape, and in the higher levels (7-9), it adds finer details about the object. Furthermore, we can interpret this observation in terms of polynomial order. Initially, it only approximates low-order polynomials and represents only basic shapes. However, at higher levels, it approximates higher-order polynomials representing finer details of the image.

Extrapolation: The INR model is a continuous function of the coordinate location; hence we extrapolate the image by feeding the pixel location outside the conventional image boundary. Our Poly-INR model is trained to generate images on the coordinate grid defined by $[0, 1]^2$. We feed the grid size $[-0.25, 1.25]^2$ to the synthesis network to generate the extrapolated images. Fig. 6.6 shows a few examples of extrapolated images. In the figure, the region within the yellow square represents the conventional coordinate grid $[0, 1]^2$. The figure shows that our INR model not only generates a continuous image outside the boundary but also preserves the geometry of the object present within the yellow square. However, in some cases, the model generates a black or white image border, resulting from the image border present in some real images of the training set.

Table 6.3: FID Score (Lower the Better) Evaluated at 512×512 for Models Trained at a Lower Resolution and Compared Against Classical Interpolation-based Upsampling.

Training Resolution	**Nearest Neighbour**	**Bilinear**	**Bicubic**	**Poly-INR**
32×32	184.39	112.28	73.86	65.15
64×64	89.24	72.41	42.97	36.30

Sampling at higher-resolution: Another advantage of using our model is the flexibility to generate images at any resolution, even if the model is trained on a lower resolution. We generate a higher-resolution image by sampling a dense coordinate grid within the $[0, 1]^2$ range. Table 6.3 shows the FID score evaluated at 512×512 for models trained on the lower-resolution ImageNet dataset. We compare the quality of upsampled images generated by our model against the classical interpolation-based upsampling methods. The table shows that our model generates crisper upsampled images, achieving a significantly better FID score than the classical interpolation-based upsampling method. However, we do not observe significant FID score improvement for our Poly-INR model trained on 128×128 or higher resolution against the classical interpolation techniques. This could be due to the limitations of the ImageNet dataset, which primarily consists of lower-resolution images than the 512×512. We used bilinear interpolation to prepare the training dataset at 512×512. As per our knowledge, there are currently no large and diverse datasets like ImageNet with high-resolution images. We believe this performance can be improved when the model has access to higher-resolution images for training. We also compare the upsampling performance with other INR-based GANs by reporting the FID scores at 1024×1024 for models trained on FFHQ-256×256 as follows: **Poly-INR:13.69, INR-GAN: 18.51, CIPS:29.59**. Our Poly-INR model provides better high-resolution sampling than the other two INR-based generators.

Interpolation: Fig. 6.7 shows that our model generates smooth interpolation between two randomly sampled images. In the first two rows of the figure, we interpolate in the latent space, and in the last two rows, we directly interpolate between the affine parameters. In our synthesis network, only the affine parameters depend on the image, and other parameters are fixed for every image. Hence interpolating in affine parameters space means interpolation in INR space. Our model provides

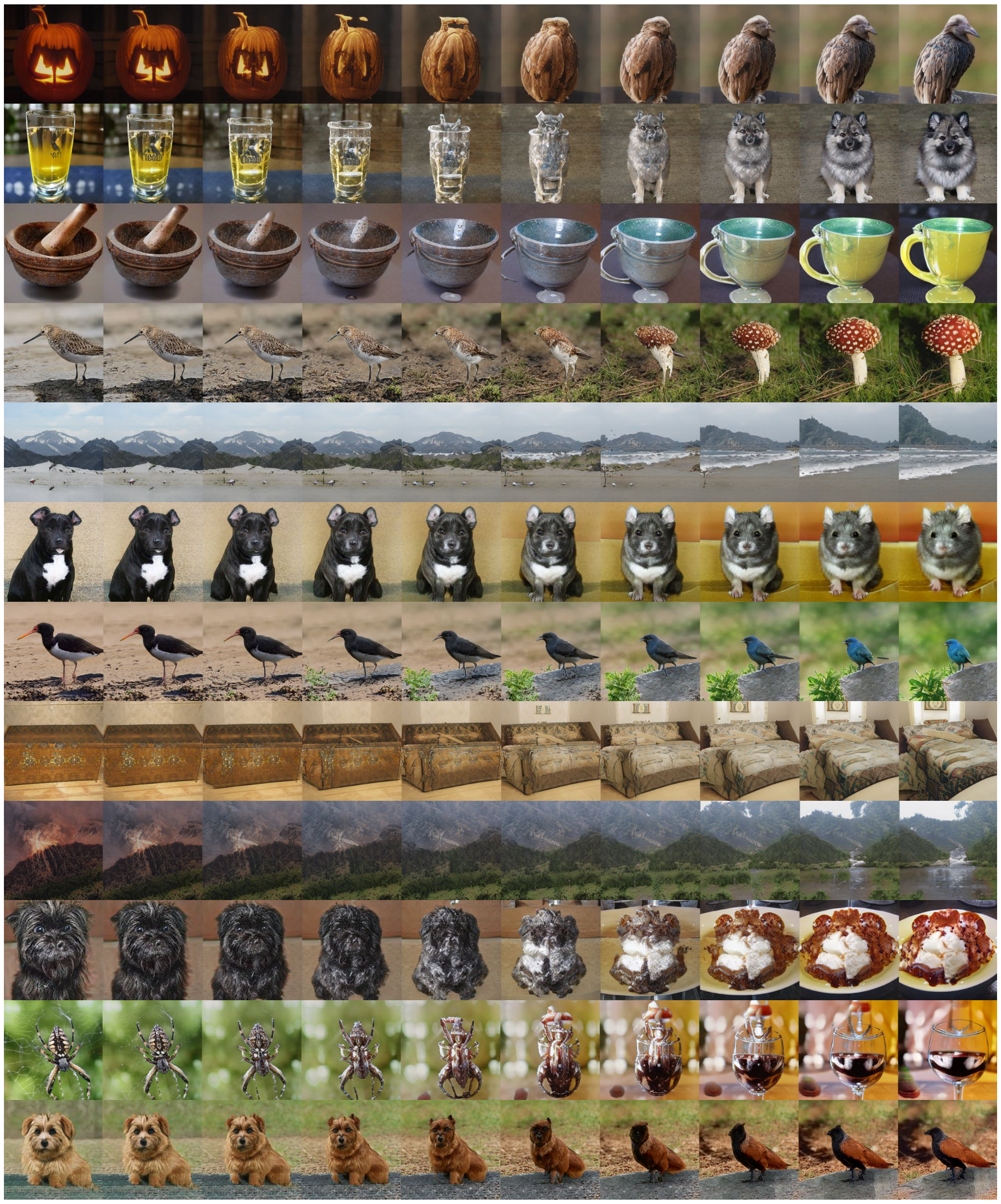


Figure 6.7: Linear Interpolation Between Two Random Points. Poly-INR Provides Smooth Interpolation Even in a High Dimension of Affine Parameters. Our Model Generates High-Fidelity Images Similar to State-Of-The-Art Models Like StyleGAN-XL but Without the Need for Convolution or Self-Attention Mechanism.

smoother interpolation even in the affine parameters space and interpolates with the geometrically coherent movement of different object parts. For example, in the first row, the eyes, nose, and mouth move systematically with the whole face.

Style-mixing: Similar to StyleGANs, our Poly-INR model transfers the style of one image to another. Our model generates smooth style mixing even though we do not use any style-mixing regularization during the training. Fig. 6.8 shows examples of style-mixing from source A to source B images. For style mixing, we first obtain the affine parameters corresponding to the source A and B images and then copy the affine parameters of A to B at various levels of the synthesis network. Copying affine parameters to higher levels (8 and 9) leads to finer changes in the style, while copying to middle levels (7, 6, and 5) leads to the coarse style change. Mixing the affine parameters at initial levels changes the shape of the generated object. In the figure, we observe that our model provides smooth style mixing while preserving the original shape of the source B object. Fig. 6.9 shows affine parameters mixing at initial levels (0-5). In the figure, we observe that copying the affine parameters at these levels changes the shape of the source B image to the source A image.

Inversion: Embedding a given image into the latent space of the GAN is an essential step for image manipulation. In our Poly-INR model, for inversion, we optimize the affine parameters to minimize the reconstruction loss, keeping the synthesis network's parameters fixed. We use VGG feature-based perceptual loss for optimization. We embed the ImageNet validation set in the affine parameters space for the quantitative evaluation. Our Poly-INR method effectively embeds images with high PSNR scores (**PSNR:26.52** and **SSIM:0.76**), better than StyleGAN-XL (PSNR:13.5 and SSIM:0.33). However, our affine parameters dimension is much larger than the StyleGAN-XL's latent space. Even though the dimension of the affine parameters is much higher, the Poly-INR model provides smooth interpolation for the embedded

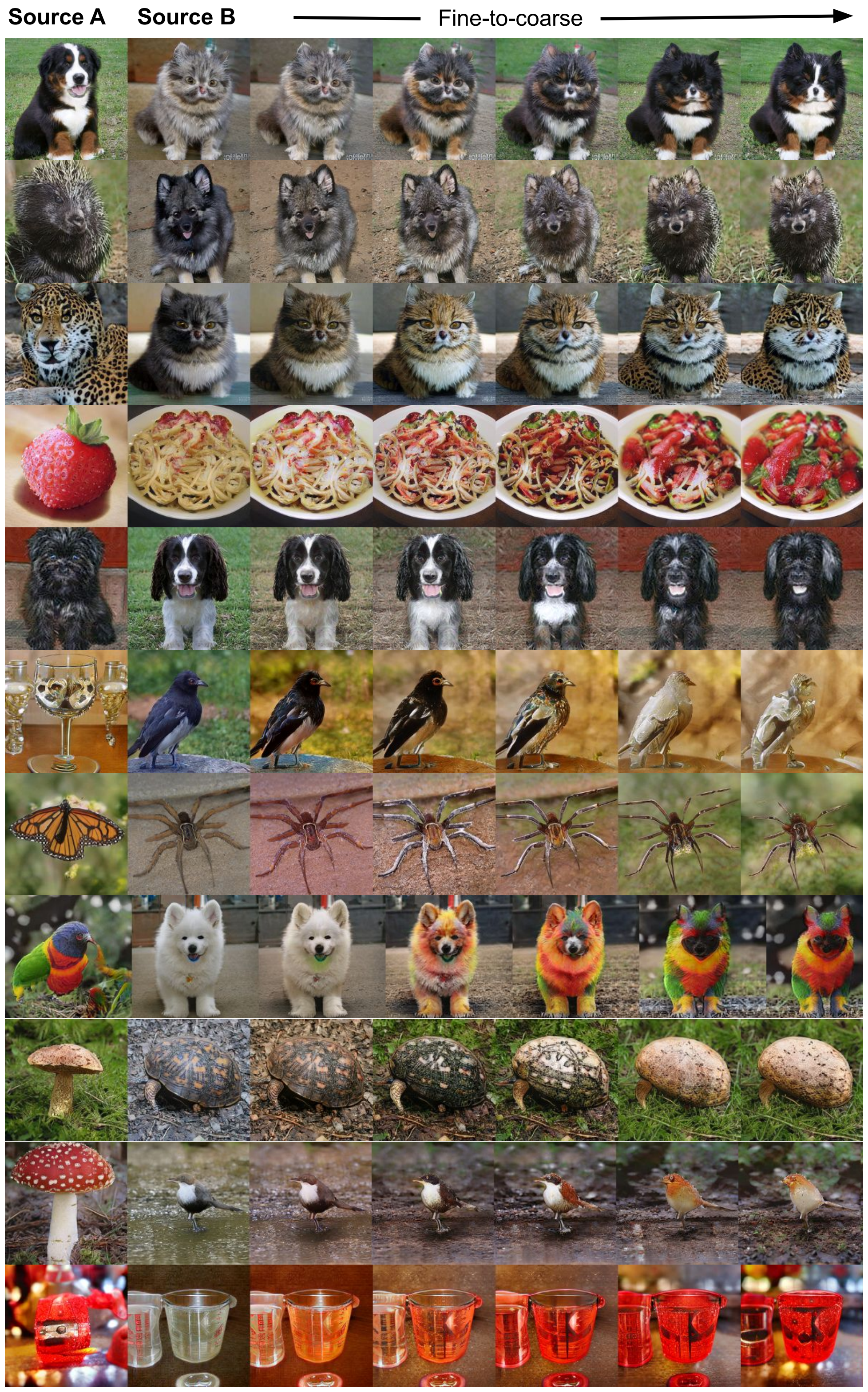


Figure 6.8: Source A and B Images Are Generated Corresponding to Random Latent Codes, and the Rest of the Images are Generated by Copying the Affine Parameters of Source A to Source B at Different Levels. Copying the Higher Levels' (8 And 9) Affine Parameters Leads to Finer Style Changes, Whereas Copying the Middle Levels' (7, 6, and 5) Leads to Coarse Style Changes.

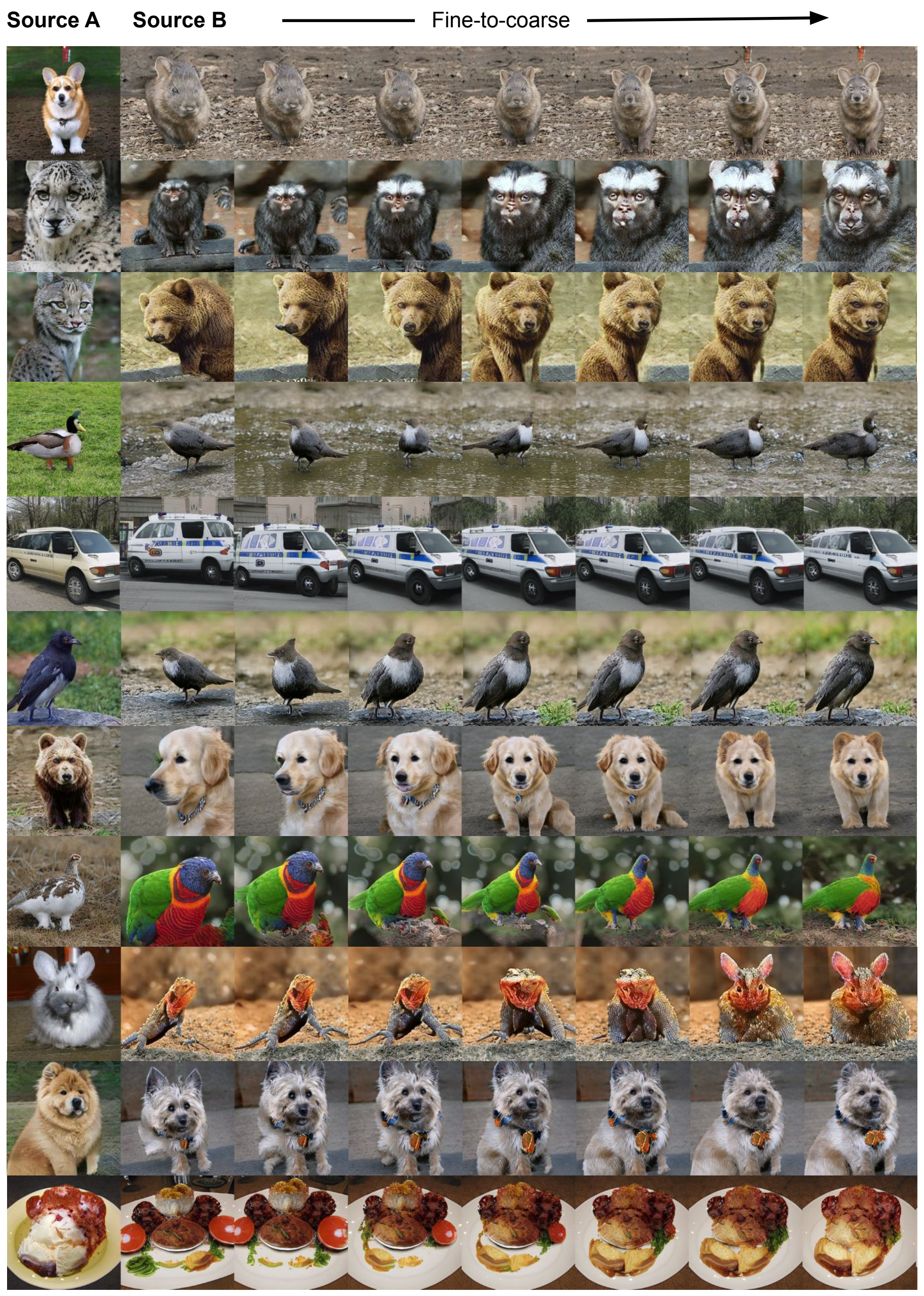


Figure 6.9: Source A and B Images Are Generated From Random Latent Codes, and Remaining Images are Generated by Copying the Affine Parameters of Source A to Source B at Different Levels. Copying the Initial Levels' (0, 1, and 2) Affine Parameters Leads to Finer Shape Changes, Whereas Copying Slightly Higher Levels' (3, 4, and 5) Leads to Coarse Shape Changes.

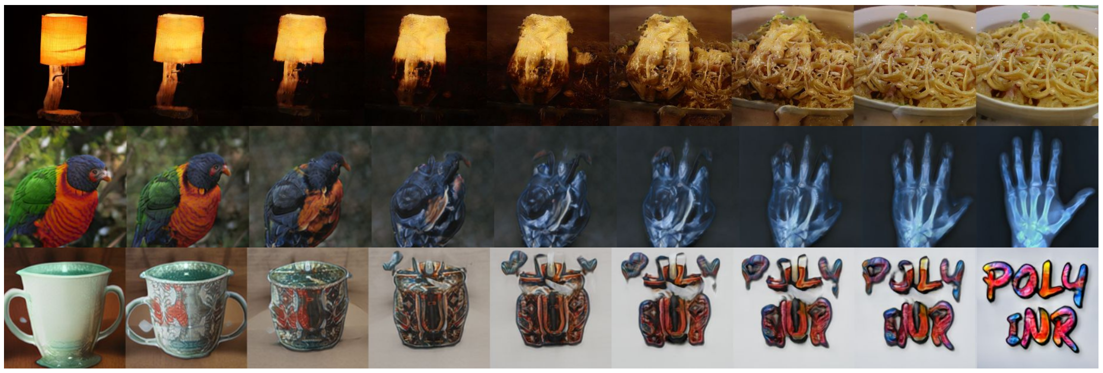


Figure 6.10: The Poly-INR Model Generates Smooth Interpolation with Embedded Images in Affine Parameters Space. The Leftmost Image (First Row) is from the ImageNet Validation Set, and the Last Two (Rightmost) Are the OOD Images.

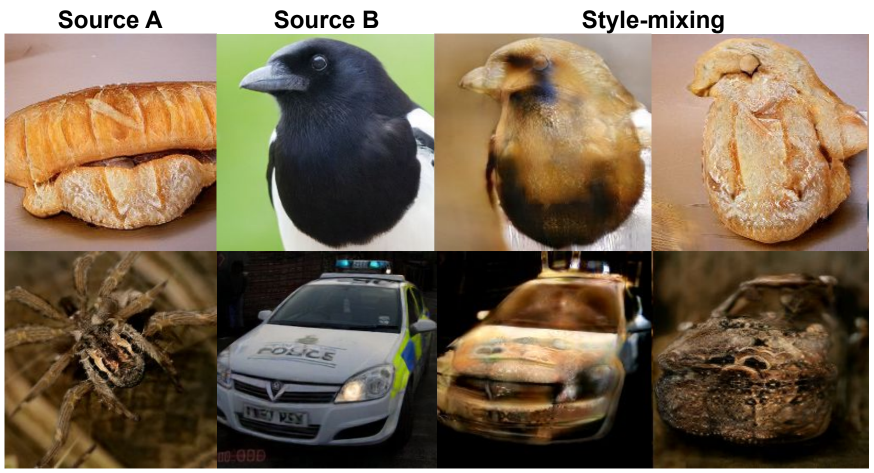


Figure 6.11: Style-Mixing with Embedded Images in Affine Parameters Space. Source B Is the Embedded Image from the Imagenet Validation Set, Mixed with the Style of Randomly Sampled Source A Image.

image. Fig. 6.10 shows examples of interpolation with embedded images. In the figure, the first row (leftmost) is the embedded image from Val set, and the last two rows (rightmost) are the out-of-distribution images. Surprisingly, our model provides smooth interpolation for OOD images. In addition, Fig. 6.11 shows smooth style-mixing with the embedded images. In some cases, we observe that the fidelity of the interpolated or style-mixed image with the embedded image is slightly less compared to samples from the training distribution. This is due to the large dimension of the embedding space, which sometimes makes the embedded point farther from the training distribution. It is possible to improve interpolation quality further by using the recently proposed pivotal tuning inversion method [116], which finetunes the generator's parameters around the embedded point.

6.4.3 Discussion

The proposed Poly-INR model performs comparably to state-of-the-art generative models on large ImageNet datasets without using convolution or self-attention layers. In addition to smooth interpolation and style-mixing, the Poly-INR model provides attractive flexibilities like image extrapolation and high-resolution sampling. In this work, while we use our INR model for 2D image datasets, it can be extended to other modalities like 3D datasets.

Challenges: One of the challenges in our INR method is the higher computation cost compared to the CNN-based generator model for high-resolution image synthesis. The INR method generates each pixel independently; hence all the computation takes place at the same resolution. In contrast, a CNN-based generator uses a multi-scale generation pipeline, making the model computationally efficient. In addition, we observe common GAN artifacts in some generated images. For example, in some cases, it generates multiple heads and limbs, missing limbs, or the object's geometry is not correctly synthesized. We suspect that the CNN-based discriminator only discriminates based on the object's parts and fails to incorporate the entire shape.

6.5 Conclusion

In this work, we propose polynomial function based implicit neural representations for large image datasets while only using Linear and ReLU layers. Our Poly-INR model captures high-frequency information and performs comparably to the state-of-the-art CNN-based generative models without using convolution, normalization, upsampling, or self-attention layers. The Poly-INR model outperforms previously proposed positional embedding-based INR GAN models. We demonstrate the effectiveness of the proposed model for various tasks like interpolation, style-mixing, extrapolation, high-resolution sampling, and image inversion. Additionally, it would be an exciting avenue for future work to extend our Poly-INR method for 3D-aware image synthesis on large datasets like ImageNet.

Chapter 7

DISCUSSION AND FUTURE WORK

In this dissertation, I presented several novel methods to build robust and controllable generative models using tools from physics, probability, and geometry. The proposed approaches improve the quality and variety of the generated samples and their controllability, all without necessarily depending on extensive training datasets. I tackled challenges related to limited data and controllability by combining physics-based rendering with generative models. I used geometric moments and non-parametric priors to make the models more robust, shape aware, reducing the need for large trainable parameters and improving the quality of interpolated images.

In Chapter 3, I explore a hybrid approach to enhance the controllability of GAN models for data augmentation by incorporating a physics-based rendering tool. Traditional GAN models like style transfer GAN or cycle GAN, while capable of generating realistic data, require extensive training datasets for diverse data generation. Collecting such datasets is often expensive and time-consuming in an industrial setting. Furthermore, controlling specific data properties with these models is challenging due to the non-disentangled and non-interpretable nature of the latent space. To address these issues, I propose a hybrid approach that combines a 3D rendering tool with a smaller GAN model, enabling more control over data properties. Certain data properties, such as shape and structure, are modeled using a few controllable parameters in the rendering tool. This enables us to generate a diverse set of images while simultaneously integrating domain knowledge directly into the modeling process, bypassing the need to learn these aspects from the training data. In contrast to conventional GAN models that generate data from scratch, our GAN model is only used to refine or apply textures to the rendered data. This method requires a significantly smaller training dataset to generate realistic and diverse data, making it more practical when few training images are available. I assessed the quality of the generated data by utilizing it for training a classification model. Our approach significantly enhances the classification model's performance in scenarios with limited training data.

In Chapter 4, I enhance the fidelity of generated images for interpolation by introducing a non-parametric prior for the latent space. Standard GAN models typically use Gaussian or uniform priors for the latent space. While these priors are easy to use, they often suffer from distribution mismatch issue, where the prior distribution diverges from the interpolated point distribution. The GAN is trained to generate realistic images based on prior points, but it traverses through low-density regions during interpolation, leading to a dip in image fidelity. To rectify this, I propose a generalized solution based on basic probability theory and optimization for addressing distribution mismatch during interpolation in GANs. Both qualitative and quantitative results showcase the effectiveness of the approach compared to standard priors. Often, the proposed method emerges as the best, and even when it doesn't, it closely matches the performance of the best alternative. The proposed non-parametric prior enhances image quality without necessitating additional training data or increased architectural complexity.

In Chapters 5 and 6, I propose a geometric moments-based architecture for the discriminator and the generator to enhance shape awareness and robustness in generative models. The discriminator is pivotal in training generative models, as it guides the generator to produce realistic images. Nevertheless, most discriminator architectures are based on CNN layers, which have a known bias towards texture rather than shape. In Chapter 5, I propose a Deep Geometric Moment (DGM) model that captures shape-related information in an end-to-end learnable manner. This model boosts the interpretability of features and effectively learns discriminative features. Both qualitative and quantitative results from standard image classification and segmentation datasets demonstrate the superior performance of our method over the corresponding baseline and standard ResNet models. Furthermore, the DGM model offers improved interpretability at various levels and facilitates easy fine-tuning on any given dataset. It also efficiently captures an object's shape under extreme affine and color aberrations, outperforming existing methodologies.

I further extended the idea of the geometric moment for the generator architecture in Chapter 6. Reconstructing images from moments assumes each pixel as a polynomial function of their coordinate location. This polynomial function is modeled using a neural network, termed as Polynomial Implicit Neural Representation (Poly-INR). In this work, I introduce a polynomial function-based implicit neural representation for large image datasets, employing only Linear and ReLU layers. The Poly-INR model successfully captures high-frequency information and rivals the performance of state-of-the-art CNN-based generative models. This is achieved without the need for convolution, normalization, upsampling, or self-attention layers. The proposed Poly-INR model surpasses the performance of previously proposed positional embedding-based INR GAN models. This chapter illustrates the proposed model's efficacy across various tasks, such as interpolation, style-mixing, extrapolation, high-resolution sampling, and image inversion.

7.1 Future Work

This dissertation prompts many intriguing opportunities for future exploration and research, as I will outline in the following discussion.

Polynomial implicit neural representation for 3D neural rendering: Inspired by the accomplishments of Neural Radiance Fields (NeRF) [98], there has been a large volume of work on 3D scene representation from 2D images [160, 163, 134, 96, 109, 62, 9]. Recently, several works have adopted NeRF as a 3D backbone for their generative models [43, 23, 87], which enables the rendering of 3D scenes as parameterized 3D volumes. Such methods typically optimize 3D representations by randomly sampling viewpoints and rendering photorealistic 2D images. This process assists in creating novel viewpoints of the same image. Our Poly-INR work (Chapter 6) has shown promising results in synthesizing 2D images, owing to its high representation capacity and competitive performance with state-of-the-art convolutional generative models. By taking the Poly-INR approach to 3D neural rendering, we can expect the model to replicate these advantages, delivering better 3D representations. The possibilities for tasks such as volumetric rendering, shape extrapolation, and scene inversion in a 3D space are noteworthy and have significant potential to advance the field of 3D neural rendering.

Polynomial implicit neural representation with diffusion model: In recent years, diffusion models [25, 52, 158] have gained significant attention, surpassing GANs in some aspects of image generation. The core strengths of diffusion models lie in their ability to generate high-quality images, along with the advantages of a tractable training procedure, a trait that has historically been a challenge with GANs. Furthermore, diffusion models exhibit remarkable flexibility when it comes to conditioning. Moving forward, I aim to synergize the strengths of both implicit neural representation and diffusion models to form a new generative model. The goal is to harness the potential of both methodologies: Poly-INR for its adaptability, the ability to train at low-resolution - samples at a high resolution, extrapolation, and control over shape and style; diffusion models for their proficient generative mechanism and capacity to yield diverse, high-quality samples. The symbiosis of these techniques could give rise to a new class of generative models excelling in sample quality, diversity, and controllability.

Hierarchical representation learning through improved geometric moments: Building upon the insights from our Poly-INR work, wherein the image is expressed in polynomial form, thereby inherently disentangling the image into shape and style components, I aim to further explore this line of thought in future research. Our Poly-INR work found that lower-order polynomial coefficients represent shape-related information, while higher-order coefficients encapsulate finer details such as style. With this in mind, my goal is to design a comprehensive feature extractor network that hierarchically segregates features - some embodying the shape of the objects, others reflecting style-related information. This hierarchical representation learning would provide a richer, more meaningful set of features. Consequently, based on the task, we can configure our classifier to prioritize shape or style, establishing a more targeted and effective learning system.

REFERENCES

[1] Adibhatla, V. A., H.-C. Chih, C.-C. Hsu, J. Cheng, M. F. Abbod and J.-S. Shieh, "Defect detection in printed circuit boards using you-only-look-once convolutional neural networks", Electronics **9**, 9, 1547 (2020).

[2] Agustsson, E., A. Sage, R. Timofte and L. Van Gool, "Optimal transport maps for distribution preserving operations on latent spaces of generative models", in "International Conference on Learning Representations (ICLR)", (2019).

[3] Alajlan, N., M. S. Kamel and G. H. Freeman, "Geometry-based image retrieval in binary image databases", IEEE transactions on pattern analysis and machine intelligence **30**, 6, 1003–1013 (2008).

[4] Alaluf, Y., O. Tov, R. Mokady, R. Gal and A. Bermano, "Hyperstyle: Stylegan inversion with hypernetworks for real image editing", in "Proceedings of the IEEE/CVF conference on computer Vision and pattern recognition", pp. 18511–18521 (2022).

[5] Aldausari, N., A. Sowmya, N. Marcus and G. Mohammadi, "Video generative adversarial networks: a review", ACM Computing Surveys (CSUR) **55**, 2, 1–25 (2022).

[6] Anokhin, I., K. Demochkin, T. Khakhulin, G. Sterkin, V. Lempitsky and D. Korzhenkov, "Image generators with conditionally-independent pixel synthesis", in "Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition", pp. 14278–14287 (2021).

[7] Arjovsky, M. and L. Bottou, "Towards principled methods for training generative adversarial networks", in "International Conference on Learning Representations (ICLR)", (2017).

[8] Arjovsky, M., S. Chintala and L. Bottou, "Wasserstein generative adversarial networks", in "International Conference on Machine Learning (ICML)", pp. 214–223 (PMLR, 2017).

[9] Barron, J. T., B. Mildenhall, D. Verbin, P. P. Srinivasan and P. Hedman, "Mip-nerf 360: Unbounded anti-aliased neural radiance fields", in "Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition", pp. 5470–5479 (2022).

[10] Bazarbaev, M., T. Chuluunsaikhan, H. Oh, G.-A. Ryu, A. Nasridinov and K.-H. Yoo, "Generation of time-series working patterns for manufacturing high-quality products through auxiliary classifier generative adversarial network", Sensors **22**, 1, 29 (2021).

[11] Bhatt, P. M., R. K. Malhan, P. Rajendran, B. C. Shah, S. Thakar, Y. J. Yoon and S. K. Gupta, "Image-based surface defect detection using deep learning: A review", Journal of Computing and Information Science in Engineering **21**, 4 (2021).

[12] Brock, A., J. Donahue and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis", in "International Conference on Learning Representations", (2018).

[13] Chen, L.-C., M. D. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam and J. Shlens, "Searching for efficient multi-scale architectures for dense image prediction", in "NeurIPS", (2018).

[14] Chen, L.-C., G. Papandreou, I. Kokkinos, K. Murphy and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs", IEEE Transactions on Pattern Analysis and Machine Intelligence **40**, 4, 834–848 (2017).

[15] Chen, L.-C., Y. Zhu, G. Papandreou, F. Schroff and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation", in "Proceedings of the European conference on computer vision (ECCV)", pp. 801–818 (2018).

[16] Chen, X., Y. Duan, R. Houthooft, J. Schulman, I. Sutskever and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets", in "Proceedings of the 30th International Conference on Neural Information Processing Systems", pp. 2180–2188 (2016).

[17] Chen, Y., S. Liu and X. Wang, "Learning continuous image representation with local implicit image function", in "Proceedings of the IEEE/CVF conference on computer vision and pattern recognition", pp. 8628–8638 (2021).

[18] Chng, S.-F., S. Ramasinghe, J. Sherrah and S. Lucey, "Gaussian activated neural radiance fields for high fidelity reconstruction and pose estimation", in "Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII", pp. 264–280 (Springer, 2022).

[19] Chong, C.-W., P. Raveendran and R. Mukundan, "Translation and scale invariants of legendre moments", Pattern recognition **37**, 1, 119–129 (2004).

[20] Cordts, M., M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth and B. Schiele, "The cityscapes dataset for semantic urban scene understanding", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 3213–3223 (2016).

[21] Dai, B., S. Fidler, R. Urtasun and D. Lin, "Towards diverse and natural image descriptions via a conditional gan", in "IEEE International Conference on Computer Vision (ICCV)", pp. 2989–2998 (2017).

[22] Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database", in "IEEE Conference on Computer Vision and Pattern Recognition (CVPR)", pp. 248–255 (2009).

[23] Deng, K., G. Yang, D. Ramanan and J.-Y. Zhu, "3d-aware conditional image synthesis", in "Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition", pp. 4434–4445 (2023).

[24] Denton, E. L., S. Chintala, R. Fergus *et al.*, “Deep generative image models using a laplacian pyramid of adversarial networks”, in “Advances in neural information processing systems”, pp. 1486–1494 (2015).

[25] Dhariwal, P. and A. Nichol, “Diffusion models beat gans on image synthesis”, Advances in Neural Information Processing Systems **34**, 8780–8794 (2021).

[26] Dinh, L., D. Krueger and Y. Bengio, “Nice: Non-linear independent components estimation”, arXiv preprint arXiv:1410.8516 (2014).

[27] Dosovitskiy, A., L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale”, in “International Conference on Learning Representations”, (2020).

[28] Elad, A. and R. Kimmel, “On bending invariant signatures for surfaces”, IEEE Transactions on pattern analysis and machine intelligence **25**, 10, 1285–1295 (2003).

[29] Everingham, M., S. Eslami, L. Van Gool, C. K. Williams, J. Winn and A. Zisserman, “The pascal visual object classes challenge: A retrospective”, International journal of computer vision **111**, 1, 98–136 (2015).

[30] Farnsworth, M., D. Tiwari, Z. Zhang, G. W. Jewell and A. Tiwari, “Augmented classification for electrical coil winding defects”, The International Journal of Advanced Manufacturing Technology **119**, 11, 6949–6965 (2022).

[31] Flusser, J., J. Boldys and B. Zitová, “Moment forms invariant to rotation and blur in arbitrary number of dimensions”, IEEE Transactions on Pattern Analysis and Machine Intelligence **25**, 2, 234–246 (2003).

[32] Flusser, J. and T. Suk, “Pattern recognition by affine moment invariants”, Pattern recognition **26**, 1, 167–174 (1993).

[33] Flusser, J., B. Zitova and T. Suk, *Moments and moment invariants in pattern recognition* (John Wiley & Sons, 2009).

[34] Fons, E., A. Sztrajman, Y. El-Laham, A. Iosifidis and S. Vyetrenko, “Hypertime: Implicit neural representations for time series”, in “NeurIPS 2022 Workshop on Synthetic Data for Empowering ML Research”, (2022).

[35] Foulonneau, A., P. Charbonnier and F. Heitz, “Affine-invariant geometric shape priors for region-based active contours”, IEEE transactions on pattern analysis and machine intelligence **28**, 8, 1352–1357 (2006).

[36] Fu, X., S. Zhang, T. Chen, Y. Lu, L. Zhu, X. Zhou, A. Geiger and Y. Liao, “Panoptic nerf: 3d-to-2d label transfer for panoptic urban scene segmentation”, in “International Conference on 3D Vision (3DV)”, (2022).

[37] Gao, C., A. Saraf, J. Kopf and J.-B. Huang, "Dynamic view synthesis from dynamic monocular video", in "Proceedings of the IEEE/CVF International Conference on Computer Vision", pp. 5712–5721 (2021).

[38] Gatys, L. A., A. S. Ecker and M. Bethge, "Image style transfer using convolutional neural networks", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)", pp. 2414–2423 (2016).

[39] Geirhos, R., P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann and W. Brendel, "Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness", in "International Conference on Learning Representations", (2018).

[40] Girshick, R., J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 580–587 (2014).

[41] Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial nets", Advances in Neural Information Processing Systems (NeuRIPS) **27** (2014).

[42] Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial networks", Communications of the ACM **63**, 11, 139–144 (2020).

[43] Gu, J., L. Liu, P. Wang and C. Theobalt, "Stylenerf: A style-based 3d-aware generator for high-resolution image synthesis", arXiv preprint arXiv:2110.08985 (2021).

[44] Gulrajani, I., F. Ahmed, M. Arjovsky, V. Dumoulin and A. C. Courville, "Improved training of wasserstein gans", in "Advances in Neural Information Processing Systems", pp. 5767–5777 (2017).

[45] Haddad, B., L. Karam, J. Ye, N. Patel and M. Braun, "Multi-feature sparse-based defect detection and classification in semiconductor units", in "IEEE International Conference on Image Processing (ICIP)", pp. 754–758 (IEEE, 2016).

[46] Hall, P., J. S. Marron and A. Neeman, "Geometric representation of high dimension, low sample size data", Journal of the Royal Statistical Society: Series B (Statistical Methodology) **67**, 3, 427–444 (2005).

[47] Härkönen, E., A. Hertzmann, J. Lehtinen and S. Paris, "Ganspace: Discovering interpretable gan controls", Advances in Neural Information Processing Systems **33**, 9841–9850 (2020).

[48] He, K., G. Gkioxari, P. Dollár and R. Girshick, "Mask r-cnn", in "Proceedings of the IEEE international conference on computer vision", pp. 2961–2969 (2017).

[49] He, K., X. Zhang, S. Ren and J. Sun, “Deep residual learning for image recognition”, in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)”, pp. 770–778 (2016).

[50] Hendrycks, D. and T. Dietterich, “Benchmarking neural network robustness to common corruptions and perturbations”, in “International Conference on Learning Representations”, (2018).

[51] Heusel, M., H. Ramsauer, T. Unterthiner, B. Nessler and S. Hochreiter, “Gans trained by a two time-scale update rule converge to a local nash equilibrium”, in “Advances in Neural Information Processing Systems”, pp. 6626–6637 (2017).

[52] Ho, J., A. Jain and P. Abbeel, “Denoising diffusion probabilistic models”, Advances in neural information processing systems **33**, 6840–6851 (2020).

[53] Ho, J., C. Saharia, W. Chan, D. J. Fleet, M. Norouzi and T. Salimans, “Cascaded diffusion models for high fidelity image generation.”, J. Mach. Learn. Res. **23**, 47–1 (2022).

[54] Hoffman, J., E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros and T. Darrell, “Cycada: Cycle-consistent adversarial domain adaptation”, in “International conference on machine learning”, pp. 1989–1998 (Pmlr, 2018).

[55] Honarvar, B., R. Paramesran and C.-L. Lim, “Image reconstruction from a complete set of geometric and complex moments”, Signal Processing **98**, 224–232 (2014).

[56] Howard, A. G., M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications”, arXiv preprint arXiv:1704.04861 (2017).

[57] Hu, M.-K., “Visual pattern recognition by moment invariants”, IRE transactions on information theory **8**, 2, 179–187 (1962).

[58] Huang, R., J. Gu, X. Sun, Y. Hou and S. Uddin, “A rapid recognition method for electronic components based on the improved YOLO-V3 network”, Electronics **8**, 8, 825 (2019).

[59] Huszár, F., “How (not) to train your generative model: Scheduled sampling, likelihood, adversary?”, arXiv preprint arXiv:1511.05101 (2015).

[60] Ioffe, S. and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift”, in “International conference on machine learning”, pp. 448–456 (PMLR, 2015).

[61] Isola, P., J.-Y. Zhu, T. Zhou and A. A. Efros, “Image-to-image translation with conditional adversarial networks”, in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)”, pp. 1125–1134 (2017).

[62] Jain, A., M. Tancik and P. Abbeel, “Putting nerf on a diet: Semantically consistent few-shot view synthesis”, in “Proceedings of the IEEE/CVF International Conference on Computer Vision”, pp. 5885–5894 (2021).

[63] Jiang, Y., S. Chang and Z. Wang, “Transgan: Two pure transformers can make one strong gan, and that can scale up”, Advances in Neural Information Processing Systems **34**, 14745–14758 (2021).

[64] Johnson, J., A. Alahi and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution”, in “European Conference on Computer Vision (ECCV)”, pp. 694–711 (Springer, 2016).

[65] Joseph-Rivlin, M., A. Zvirin and R. Kimmel, “Momen (e) t: Flavor the moments in learning to classify shapes”, in “Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops”, pp. 0–0 (2019).

[66] Karras, T., T. Aila, S. Laine and J. Lehtinen, “Progressive growing of gans for improved quality, stability, and variation”, in “International Conference on Learning Representations (ICLR)”, (2018).

[67] Karras, T., M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen and T. Aila, “Alias-free generative adversarial networks”, Advances in Neural Information Processing Systems **34**, 852–863 (2021).

[68] Karras, T., S. Laine and T. Aila, “A style-based generator architecture for generative adversarial networks”, in “Proceedings of the IEEE/CVF conference on computer vision and pattern recognition”, pp. 4401–4410 (2019).

[69] Karras, T., S. Laine, M. Aittala, J. Hellsten, J. Lehtinen and T. Aila, “Analyzing and improving the image quality of stylegan”, in “Proceedings of the IEEE/CVF conference on computer vision and pattern recognition”, pp. 8110–8119 (2020).

[70] Khotanzad, A. and Y. H. Hong, “Invariant image recognition by zernike moments”, IEEE Transactions on pattern analysis and machine intelligence **12**, 5, 489–497 (1990).

[71] Kilcher, Y., A. Lucchi and T. Hofmann, “Semantic interpolation in implicit models”, in “International Conference on Learning Representations (ICLR)”, (2018).

[72] Kim, H. S. and H.-K. Lee, “Invariant image watermark using zernike moments”, IEEE transactions on Circuits and Systems for Video Technology **13**, 8, 766–775 (2003).

[73] Kingma, D. P. and M. Welling, “Auto-encoding variational bayes”, in “International Conference on Learning Representations (ICLR)”, (2014).

[74] Kingma, D. P. and M. Welling, “Stochastic gradient vb and the variational auto-encoder”, in “Second International Conference on Learning Representations, ICLR”, vol. 19, p. 121 (2014).

[75] Krizhevsky, A. and G. Hinton, "Learning multiple layers of features from tiny images", Tech. rep., University of Toronto (2009).

[76] Krizhevsky, A., I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", Advances in neural information processing systems **25**, 1097–1105 (2012).

[77] Kumar, A. and R. Chellappa, "Disentangling 3D Pose in A Dendritic CNN for Unconstrained 2D Face Alignment", in "IEEE Conference on Computer Vision and Pattern Recognition (CVPR)", pp. 430–439 (2018).

[78] Kynkäänniemi, T., T. Karras, S. Laine, J. Lehtinen and T. Aila, "Improved precision and recall metric for assessing generative models", Advances in Neural Information Processing Systems **32** (2019).

[79] LeCun, Y., Y. Bengio and G. Hinton, "Deep learning", Nature **521**, 7553, 436–444 (2015).

[80] Ledig, C., L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)", pp. 4681–4690 (2017).

[81] Lee, K., H. Chang, L. Jiang, H. Zhang, Z. Tu and C. Liu, "Vitgan: Training gans with vision transformers", in "International Conference on Learning Representations", (2021).

[82] Leśniak, D., I. Sieradzki and I. Podolak, "Distribution-interpolation trade off in generative models", in "International Conference on Learning Representations (ICLR)", (2019).

[83] Li, C.-L., W.-C. Chang, Y. Cheng, Y. Yang and B. Póczos, "Mmd gan: Towards deeper understanding of moment matching network", Advances in neural information processing systems **30** (2017).

[84] Li, D., D. Chen, B. Jin, L. Shi, J. Goh and S.-K. Ng, "Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks", in "International conference on artificial neural networks", pp. 703–716 (Springer, 2019).

[85] Li, D., X. Shen, Y. Yu, H. Guan, H. Wang and D. Li, "Ggm-net: Graph geometric moments convolution neural network for point cloud shape classification", IEEE Access **8**, 124989–124998 (2020).

[86] Li, R., X. Li, C.-W. Fu, D. Cohen-Or and P.-A. Heng, "Pu-gan: a point cloud upsampling adversarial network", in "Proceedings of the IEEE/CVF international conference on computer vision", pp. 7203–7212 (2019).

[87] Li, S., J. van de Weijer, Y. Wang, F. S. Khan, M. Liu and J. Yang, "3d-aware multi-class image-to-image translation with nerfs", in "Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition", pp. 12652–12662 (2023).

[88] Lim, J. H. and J. C. Ye, "Geometric gan", arXiv preprint arXiv:1705.02894 (2017).

[89] Liu, H., Z. Wan, W. Huang, Y. Song, X. Han and J. Liao, "Pd-gan: Probabilistic diverse gan for image inpainting", in "Proceedings of the IEEE/CVF conference on computer vision and pattern recognition", pp. 9371–9381 (2021).

[90] Liu, Y., T. Han, S. Ma, J. Zhang, Y. Yang, J. Tian, H. He, A. Li, M. He, Z. Liu *et al.*, "Summary of chatgpt/gpt-4 research and perspective towards the future of large language models", arXiv preprint arXiv:2304.01852 (2023).

[91] Liu, Y., F. Wei, J. Shao, L. Sheng, J. Yan and X. Wang, "Exploring disentangled feature representation beyond face identification", in "IEEE Conference on Computer Vision and Pattern Recognition (CVPR)", pp. 2080–2089 (2018).

[92] Liu, Y., Y. Yeh, T. Fu, S. Wang, W. Chiu and Y. F. Wang, "Detach and Adapt: Learning Cross-Domain Disentangled Deep Representation", in "IEEE Conference on Computer Vision and Pattern Recognition (CVPR)", pp. 8867–8876 (2018).

[93] Liu, Z., P. Luo, X. Wang and X. Tang, "Deep learning face attributes in the wild", in "Proceedings of the IEEE International Conference on Computer Vision", pp. 3730–3738 (2015).

[94] Luciano, L. and A. B. Hamza, "Deep learning with geodesic moments for 3d shape classification", Pattern Recognition Letters **105**, 182–190 (2018).

[95] Mao, X., Q. Li, H. Xie, R. Y. Lau, Z. Wang and S. Paul Smolley, "Least squares generative adversarial networks", in "Proceedings of the IEEE international conference on computer vision", pp. 2794–2802 (2017).

[96] Martin-Brualla, R., N. Radwan, M. S. Sajjadi, J. T. Barron, A. Dosovitskiy and D. Duckworth, "Nerf in the wild: Neural radiance fields for unconstrained photo collections", in "Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition", pp. 7210–7219 (2021).

[97] Mescheder, L., M. Oechsle, M. Niemeyer, S. Nowozin and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space", in "Proceedings of the IEEE/CVF conference on computer vision and pattern recognition", pp. 4460–4470 (2019).

[98] Mildenhall, B., P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis", Communications of the ACM **65**, 1, 99–106 (2021).

[99] Mirza, M. and S. Osindero, “Conditional generative adversarial nets”, arXiv preprint arXiv:1411.1784 (2014).

[100] Miyato, T., T. Kataoka, M. Koyama and Y. Yoshida, “Spectral normalization for generative adversarial networks”, arXiv preprint arXiv:1802.05957 (2018).

[101] Mou, S., M. Cao, Z. Hong, P. Huang, J. Shan and J. Shi, “Synthetic defect generation for display front-of-screen quality inspection: A survey”, arXiv preprint arXiv:2203.03429 (2022).

[102] Nash, C., J. Menick, S. Dieleman and P. Battaglia, “Generating images with sparse representations”, in “International Conference on Machine Learning”, pp. 7958–7968 (PMLR, 2021).

[103] Nowozin, S., B. Cseke and R. Tomioka, “f-gan: Training generative neural samplers using variational divergence minimization”, in “Advances in Neural Information Processing Systems”, pp. 271–279 (2016).

[104] Ntavelis, E., M. Shahbazi, I. Kastanis, R. Timofte, M. Danelljan and L. Van Gool, “Arbitrary-scale image synthesis”, in “Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition”, pp. 11533–11542 (2022).

[105] Park, K., U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz and R. Martin-Brualla, “Nerfies: Deformable neural radiance fields”, in “Proceedings of the IEEE/CVF International Conference on Computer Vision”, pp. 5865–5874 (2021).

[106] Pathak, D., P. Krahenbuhl, J. Donahue, T. Darrell and A. A. Efros, “Context encoders: Feature learning by inpainting”, in “IEEE Conference on Computer Vision and Pattern Recognition (CVPR)”, pp. 2536–2544 (2016).

[107] Peebles, W. and S. Xie, “Scalable diffusion models with transformers”, arXiv preprint arXiv:2212.09748 (2022).

[108] Plachetka, T., “Pov ray: persistence of vision parallel raytracer”, in “Proc. of Spring Conf. on Computer Graphics, Budmerice, Slovakia”, vol. 123, p. 129 (1998).

[109] Pumarola, A., E. Corona, G. Pons-Moll and F. Moreno-Noguer, “D-nerf: Neural radiance fields for dynamic scenes”, in “Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition”, pp. 10318–10327 (2021).

[110] Radford, A., L. Metz and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks”, in “International Conference on Learning Representations (ICLR)”, (2016).

[111] Ramasinghe, S. and S. Lucey, “Beyond periodicity: towards a unifying framework for activations in coordinate-mlps”, in “Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII”, pp. 142–158 (Springer, 2022).

[112] Reed, A. W., H. Kim, R. Anirudh, K. A. Mohan, K. Champley, J. Kang and S. Jayasuriya, "Dynamic ct reconstruction from limited views with implicit neural representations and parametric motion fields", in "Proceedings of the IEEE/CVF International Conference on Computer Vision", pp. 2258–2268 (2021).

[113] Reed, S., Z. Akata, X. Yan, L. Logeswaran, B. Schiele and H. Lee, "Generative adversarial text to image synthesis", in "33rd International Conference on Machine Learning", pp. 1060–1069 (2016).

[114] Reeves, A. P., R. J. Prokop, S. E. Andrews and F. P. Kuhl, "Three-dimensional shape analysis using moments and fourier descriptors", IEEE Transactions on Pattern Analysis and Machine Intelligence **10**, 6, 937–943 (1988).

[115] Ren, S., K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks", Advances in Neural Information Processing Systems (NeuRIPS) **28** (2015).

[116] Roich, D., R. Mokady, A. H. Bermano and D. Cohen-Or, "Pivotal tuning for latent-based editing of real images", ACM Transactions on Graphics (TOG) **42**, 1, 1–13 (2022).

[117] Ronneberger, O., P. Fischer and T. Brox, "U-net: Convolutional networks for biomedical image segmentation", in "International Conference on Medical image computing and computer-assisted intervention", pp. 234–241 (Springer, 2015).

[118] Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge", International journal of computer vision **115**, 3, 211–252 (2015).

[119] Sadjadi, F. A. and E. L. Hall, "Three-dimensional moment invariants", IEEE Transactions on Pattern Analysis and Machine Intelligence , 2, 127–136 (1980).

[120] Saharia, C., W. Chan, H. Chang, C. Lee, J. Ho, T. Salimans, D. Fleet and M. Norouzi, "Palette: Image-to-image diffusion models", in "ACM SIGGRAPH 2022 Conference Proceedings", pp. 1–10 (2022).

[121] Saharia, C., W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding", Advances in Neural Information Processing Systems **35**, 36479–36494 (2022).

[122] Salimans, T., I. Goodfellow, W. Zaremba, V. Cheung, A. Radford and X. Chen, "Improved techniques for training gans", in "Advances in Neural Information Processing Systems", pp. 2234–2242 (2016).

[123] Sampath, V., I. Maurtua, J. J. Aguilar Martín and A. Gutierrez, "A survey on generative adversarial networks for imbalance problems in computer vision tasks", Journal of big Data **8**, 1, 1–59 (2021).

[124] Sandler, M., A. Howard, M. Zhu, A. Zhmoginov and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks”, in “Proceedings of the IEEE conference on computer vision and pattern recognition”, pp. 4510–4520 (2018).

[125] Sauer, A., K. Chitta, J. Müller and A. Geiger, “Projected gans converge faster”, Advances in Neural Information Processing Systems **34**, 17480–17492 (2021).

[126] Sauer, A., T. Karras, S. Laine, A. Geiger and T. Aila, “Stylegan-t: Unlocking the power of gans for fast large-scale text-to-image synthesis”, arXiv preprint arXiv:2301.09515 (2023).

[127] Sauer, A., K. Schwarz and A. Geiger, “Stylegan-xl: Scaling stylegan to large diverse datasets”, in “ACM SIGGRAPH 2022 Conference Proceedings”, pp. 1–10 (2022).

[128] Selvaraju, R. R., M. Cogswell, A. Das, R. Vedantam, D. Parikh and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization”, in “Proceedings of the IEEE international conference on computer vision”, pp. 618–626 (2017).

[129] Shah, V. and C. Hegde, “Solving linear inverse problems using gan priors: An algorithm with provable guarantees”, in “2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)”, pp. 4609–4613 (IEEE, 2018).

[130] Shrivastava, A., T. Pfister, O. Tuzel, J. Susskind, W. Wang and R. Webb, “Learning from simulated and unsupervised images through adversarial training”, in “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)”, pp. 2107–2116 (2017).

[131] Singh, R., R. Garg, N. S. Patel and M. W. Braun, “Generative adversarial networks for synthetic defect generation in assembly and test manufacturing”, in “31st Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)”, pp. 1–5 (IEEE, 2020).

[132] Singh, R., P. Turaga, S. Jayasuriya, R. Garg and M. Braun, “Non-parametric priors for generative adversarial networks”, in “International Conference on Machine Learning (ICML)”, pp. 5838–5847 (PMLR, 2019).

[133] Sitzmann, V., J. Martel, A. Bergman, D. Lindell and G. Wetzstein, “Implicit neural representations with periodic activation functions”, Advances in Neural Information Processing Systems **33**, 7462–7473 (2020).

[134] Sitzmann, V., S. Rezchikov, B. Freeman, J. Tenenbaum and F. Durand, “Light field networks: Neural scene representations with single-evaluation rendering”, Advances in Neural Information Processing Systems **34**, 19313–19325 (2021).

[135] Sitzmann, V., M. Zollhöfer and G. Wetzstein, “Scene representation networks: Continuous 3d-structure-aware neural scene representations”, Advances in Neural Information Processing Systems **32** (2019).

[136] Skorokhodov, I., S. Ignatyev and M. Elhoseiny, “Adversarial generation of continuous images”, in “Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition”, pp. 10753–10764 (2021).

[137] Skorokhodov, I., G. Sotnikov and M. Elhoseiny, “Aligning latent and image spaces to connect the unconnectable”, in “Proceedings of the IEEE/CVF International Conference on Computer Vision”, pp. 14144–14153 (2021).

[138] Skorokhodov, I., S. Tulyakov and M. Elhoseiny, “Stylegan-v: A continuous video generator with the price, image quality and perks of stylegan2”, in “Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition”, pp. 3626–3636 (2022).

[139] Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, “Going deeper with convolutions”, in “Proceedings of the IEEE conference on computer vision and pattern recognition”, pp. 1–9 (2015).

[140] Tabernik, D., S. Šela, J. Skvarč and D. Skočaj, “Segmentation-based deep-learning approach for surface-defect detection”, Journal of Intelligent Manufacturing **31**, 3, 759–776 (2020).

[141] Tan, M. and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks”, in “International conference on machine learning”, pp. 6105–6114 (PMLR, 2019).

[142] Tancik, M., P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron and R. Ng, “Fourier features let networks learn high frequency functions in low dimensional domains”, Advances in Neural Information Processing Systems **33**, 7537–7547 (2020).

[143] Teague, M. R., “Image analysis via the general theory of moments”, Josa **70**, 8, 920–930 (1980).

[144] Theodoridis, T., K. Loumponias, N. Vretos and P. Daras, “Zernike pooling: Generalizing average pooling using zernike moments”, IEEE Access **9**, 121128–121136 (2021).

[145] Touvron, H., M. Cord, M. Douze, F. Massa, A. Sablayrolles and H. Jégou, “Training data-efficient image transformers & distillation through attention”, in “International Conference on Machine Learning”, pp. 10347–10357 (PMLR, 2021).

[146] Tov, O., Y. Alaluf, Y. Nitzan, O. Patashnik and D. Cohen-Or, “Designing an encoder for stylegan image manipulation”, ACM Transactions on Graphics (TOG) **40**, 4, 1–14 (2021).

[147] Tuceryan, M., “Moment-based texture segmentation”, Pattern recognition letters **15**, 7, 659–668 (1994).

[148] Tzeng, E., J. Hoffman, K. Saenko and T. Darrell, "Adversarial discriminative domain adaptation", in "IEEE Conference on Computer Vision and Pattern Recognition (CVPR)", vol. 1, p. 4 (2017).

[149] Van Gansbeke, W., S. Vandenhende, S. Georgoulis, M. Proesmans and L. Van Gool, "Scan: Learning to classify images without labels", in "European Conference on Computer Vision", pp. 268–285 (Springer, 2020).

[150] Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, "Attention is all you need", in "Advances in neural information processing systems", pp. 5998–6008 (2017).

[151] Wali, A., Z. Alamgir, S. Karim, A. Fawaz, M. B. Ali, M. Adan and M. Mujtaba, "Generative adversarial networks for speech processing: A review", Computer Speech & Language **72**, 101308 (2022).

[152] Wang, H., J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie and M. Guo, "Graphgan: Graph representation learning with generative adversarial nets", in "Proceedings of the AAAI conference on artificial intelligence", vol. 32 (2018).

[153] Wang, L. and G. Healey, "Using zernike moments for the illumination and geometry invariant classification of multispectral texture", IEEE Transactions on Image Processing **7**, 2, 196–203 (1998).

[154] Wang, Z., L. Yu and L. Pu, "Defect simulation in sem images using generative adversarial networks", in "Metrology, Inspection, and Process Control for Semiconductor Manufacturing XXXV", vol. 11611, pp. 113–119 (SPIE, 2021).

[155] White, T., "Sampling generative networks", arXiv preprint arXiv:1609.04468 (2016).

[156] Wu, J., S. Qiu, Y. Kong, Y. Chen, L. Senhadji and H. Shu, "Momentsnet: a simple learning-free method for binary image recognition", in "2017 IEEE International Conference on Image Processing (ICIP)", pp. 2667–2671 (IEEE, 2017).

[157] Xian, W., J.-B. Huang, J. Kopf and C. Kim, "Space-time neural irradiance fields for free-viewpoint video", in "Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition", pp. 9421–9431 (2021).

[158] Yang, L., Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, Y. Shao, W. Zhang, B. Cui and M.-H. Yang, "Diffusion models: A comprehensive survey of methods and applications", arXiv preprint arXiv:2209.00796 (2022).

[159] Yap, P.-T. and R. Paramesran, "An efficient method for the computation of legendre moments", IEEE Transactions on Pattern Analysis and Machine Intelligence **27**, 12, 1996–2002 (2005).

[160] Yariv, L., Y. Kasten, D. Moran, M. Galun, M. Atzmon, B. Ronen and Y. Lipman, "Multiview neural surface reconstruction by disentangling geometry and appearance", Advances in Neural Information Processing Systems **33**, 2492–2502 (2020).

[161] Yin, W., Y. Fu, L. Sigal and X. Xue, "Semi-latent GAN: Learning to generate and modify facial images from attributes", arXiv preprint arXiv:1704.02166 (2017).

[162] Yoon, Y., I. Chung, L. Wang and K.-J. Yoon, "Spheresr: 360deg image super-resolution with arbitrary projection via continuous spherical image representation", in "Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition", pp. 5677–5686 (2022).

[163] Yu, A., V. Ye, M. Tancik and A. Kanazawa, "pixelnerf: Neural radiance fields from one or few images", in "Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition", pp. 4578–4587 (2021).

[164] Yu, F., A. Seff, Y. Zhang, S. Song, T. Funkhouser and J. Xiao, "LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop", arXiv preprint arXiv:1506.03365 (2015).

[165] Zhang, B., S. Gu, B. Zhang, J. Bao, D. Chen, F. Wen, Y. Wang and B. Guo, "Styleswin: Transformer-based gan for high-resolution image generation", in "Proceedings of the IEEE/CVF conference on computer vision and pattern recognition", pp. 11304–11314 (2022).

[166] Zhang, H., I. Goodfellow, D. Metaxas and A. Odena, "Self-attention generative adversarial networks", arXiv preprint arXiv:1805.08318 (2018).

[167] Zhang, H., H. Shu, G. Coatrieux, J. Zhu, Q. J. Wu, Y. Zhang, H. Zhu and L. Luo, "Affine legendre moment invariants for image watermarking robust to geometric distortions", IEEE Transactions on Image Processing **20**, 8, 2189–2199 (2011).

[168] Zhang, H., H. Shu, G. N. Han, G. Coatrieux, L. Luo and J. L. Coatrieux, "Blurred image recognition by legendre moment invariants", IEEE Transactions on Image Processing **19**, 3, 596–611 (2009).

[169] Zhang, H., T. Xu, H. Li, S. Zhang, X. Wang, X. Huang and D. N. Metaxas, "StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks", in "IEEE International Conference on Computer Vision (ICCV)", pp. 5907–5915 (2017).

[170] Zhang, M., M. Qamar, T. Kang, Y. Jung, C. Zhang, S.-H. Bae and C. Zhang, "A survey on graph diffusion models: Generative ai in science for molecule, protein and material", arXiv preprint arXiv:2304.01565 (2023).

[171] Zhang, Y., Z. Gan, K. Fan, Z. Chen, R. Henao, D. Shen and L. Carin, "Adversarial feature matching for text generation", in "International conference on machine learning", pp. 4006–4015 (PMLR, 2017).

[172] Zhao, L., Z. Zhang, T. Chen, D. Metaxas and H. Zhang, “Improved transformer for high-resolution gans”, Advances in Neural Information Processing Systems **34**, 18367–18380 (2021).

[173] Zhu, J., Y. Shen, D. Zhao and B. Zhou, “In-domain gan inversion for real image editing”, in “European conference on computer vision”, pp. 592–608 (Springer, 2020).

[174] Zhu, J.-Y., P. Krähenbühl, E. Shechtman and A. A. Efros, “Generative visual manipulation on the natural image manifold”, in “European Conference on Computer Vision (ECCV)”, pp. 597–613 (Springer, 2016).

[175] Zhu, J.-Y., T. Park, P. Isola and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks”, in “Proceedings of the IEEE International Conference on Computer Vision (ICCV)”, pp. 2223–2232 (2017).